1998 0210 052

An Incompleteness Handling Methodology

for Validation of

Bayesian Knowledge Bases

THESIS
David J. Bawcom
First Lieutenant

AFIT/GCS/ENG/97D-02

DTIC QUALITY INSPECTED 4

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

AFIT/GCS/ENG/97D-02

An Incompleteness Handling Methodology

for Validation of

Bayesian Knowledge Bases

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science

David J. Bawcom, B.S.

First Lieutenant

December, 1997

AFIT/GCS/ENG/97D-02

An Incompleteness Handling Methodology

for Validation of

Bayesian Knowledge Bases

David J. Bawcom, B.S.

First Lieutenant

Approved:

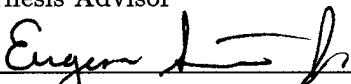_Sheila B Banks_ (signature)   18 Nov 97
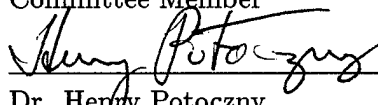
Dr. Sheila B. Banks   Date
Thesis Advisor

Eugene Santos (signature)   18 Nov 97

Dr. Eugene Santos Jr.   Date
Committee Member

Henry Potoczny (signature)   18 Nov. 97

Dr. Henry Potoczny   Date
Committee Member

Acknowledgements

Table of Contents

List of Figures

AFIT/GCS/ENG/97D-02

Abstract

The PESKI (Probabilities, Expert Systems, Knowledge, and Inference) system attempts to address some of the problems in expert system design through the use of the Bayesian Knowledge Base (BKB) representation. Knowledge gathered from a domain expert is placed into this framework and inferencing is performed over it. However, by the nature of BKBs, not all knowledge is incorporated, i.e. the representation need not be a complete representation of all combinations and possibilities of the knowledge, as this would be impractical in many real-world systems. Therefore, inherent in such a system is the problem of incomplete knowledge, or spaces within the knowledge base where areas of lacking knowledge preclude or hinder arrival at a solution. Some of this knowledge is intentionally omitted because its not needed for inferencing, while other knowledge is erroneously omitted but necessary for valid results. Intentional omission, a strength of the BKB representation, allows for capturing only the relevant portions of knowledge critical to modeling an expert's knowledge within a domain. This research proposes a method for handling the latter form of incompleteness administered through a graphical interface. The incompleteness is then able to be detected and corrected by the knowledge engineer in an intuitive fashion.

An Incompleteness Handling Methodology

for Validation of

Bayesian Knowledge Bases

I. Introduction

Knowledge based systems, also known as expert systems, are a growing area in the field of artificial intelligence. These systems exist in many different types of applications and their uses include reasoning and decision making capabilities. The knowledge contained in these types of systems often involve difficult domains in which few experts exist. By providing a knowledge based system with this type of rare intelligence, the knowledge is able to be used more often and the human origin is allowed to concentrate on more complex problems.

The first knowledge based system can be traced back to the mid 1960's. Researchers at Stanford University decided to try encoding the heuristic knowledge of an expert chemist into a system later named DENDRAL [13]. This knowledge was used to elucidate the structure of complex molecules from mass spectrograms and often outperformed human experts. The project was a success, and for the first time highlighted the fact that an intelligent computer program could be developed to emphasize what the program knew about a problem rather than on some clever search algorithm [7]. Other systems soon followed in other domains: MYCIN [34] diagnosed blood disorders in the medical field, HEARSAY [8] was created for spoken language understanding, PROSPECTOR [6] in the geology domain, and XCON [17] in manufacturing. These systems proved that constrained real-world problems using specific heuristic knowledge could be solved through applied knowledge representation and reasoning techniques. Knowledge based systems continue to flourish even today. This availability is fueled by advancing computer technology and an increasing availability of sophisticated development environments [19]. The predominant role of expert systems has been in the diagnosis arena, due mostly to the fact that it is the same role that experts most often play [7].

Developing these knowledge based systems is far from trivial. There are many facets in the construction of a complete knowledge based system. First, in the knowledge acquisition phase of a system, knowledge engineers must thoroughly extract knowledge from an expert. Methods of extracting this knowledge are numerous. A representation scheme for this knowledge must then be chosen. The knowledge engineer must carefully build this knowledge representation into an expert system for which it can be inferenced over. There are many opportunities for inputting incorrect, incomplete, or inconcise information while building a new system. Often many modifications to the knowledge base are necessary in existing systems as well, which can often adversely affect other areas unintentionally. For these reasons, and others which will be discussed in the following chapter, verification and validation (V & V) of these knowledge based systems is an increasingly important part of today's sophisticated knowledge based systems.

A great amount of research has been performed in the area of V & V over the past few years. However, with the large number of representations, inferencing techniques, and knowledge acquisition techniques, there is no common consensus on the best method of V & V. This research will center on one particular representation scheme known as Bayesian Knowledge Bases (BKBs). This BKB representation scheme is part of an overall expert system shell known as Probabilities, Expert System, Knowledge, and Inference[1] (PESKI), which is an integrated framework for expert system development [28].

This research focuses on developing a methodology to correct one problematic area of V & V, namely unintentional incompleteness that may be present in the knowledge base. The results are currently integrated into the PESKI system. The incompleteness is recognized in the validation phase, and a tool for correcting this lack of knowledge is introduced. Test cases are the instrument used for validating BKBs in PESKI. These test cases are submitted to the system and its results are compared to expected results. Incompleteness occurs when the inferencing cannot reach an expected solution as defined by a test case. This incompleteness can come from several different sources and are investigated in the following chapter. After identifying incompleteness does exist in the BKB, the knowledge

---

[1]See Appendix A for further information about PESKI.

engineer can implement within PESKI the existing data mining utility[2] for correction as well as the graphical incompleteness tool developed within this research. The graphical incompleteness tool assists the knowledge engineer in locating the area of incompleteness and then extracts the missing information from him/her for insertion into the knowledge base.

In chapter II, we provide a thorough discussion of the problem background. Chapter III discusses the methodology to identify and repair BKB incompleteness developed within this research. Chapter IV introduces the tool built within this methodology framework. Specific test-case examples and results are explored in Chapter V. Chapter VI discusses conclusions drawn from this research, while recommendations for further research are offered in Chapter VII.

---

[2]Information about the data mining utility can be found in Stein [37].

II. Problem Background

The difficulties in the development of knowledge based systems, particularly with knowledge representation and knowledge acquisition, often leads to errors in several forms. One of these types of errors is incompleteness. This research stems from a need to handle incompleteness during the validation stage of development. This chapter provides some of the necessary background material for understanding the need for this type of validation and error handling.

2.1 Verification & Validation Testing

As expert systems become more and more common as well as more critical in their application domains (e.g. medicine, air traffic control), their success will hinge upon their performance and the validity of their results. This performance will depend upon, among other things, thorough verification and validation (V & V) and, more specifically, the techniques used in performing this V & V. A knowledge based system is built for its "intelligence." If this "intelligence" is filled with mistakes leading to erroneous problem solving, or incompleteness leading to a shortfall of reasoning capability, their credibility, and the credibility of expert systems in general will decrease. Much work has been done in the last few years to address some of these issues; however, at this time there seems to be no agreed upon methodology for performing knowledge base V & V.

While the main objectives of V & V are closely knitted together, it is important to understand the distinct differences between them. Verification is best defined as making sure the system is built correctly. Critical to this step is ensuring all information deemed necessary is included and that this information is interpreted and applied correctly by the system being inspected. If specifications exist for a particular system, verification will check for compliance with these specifications. It also oversees the correct software syntax from which it was built [1]. Verification is often referred to as clear-box testing.

Validation, on the other hand, is used to ensure the output of the system is correct. It is also used to check the system developed is what the users requested. It must assume the knowledge base was built satisfactorily. Typically, expert system validation consists

of running a sequence of test cases through the system and comparing system results against known results or expert opinions [22]. This is a time-consuming process and never guarantees finding all errors, especially in larger systems. O'Keefe et al. stated "Validation can be considered the cornerstone of evaluation (of an expert system), since highly efficient implementations of invalid systems are useless" [22]. Validation is often referred to as black-box testing. Concern is placed not upon what is inside the system, but what the results are coming out of the system. Despite the importance of validation, the majority of V & V literature is solely concerned with verification, specifically automatic rule-based error checking. This aspect of V & V has now become reasonably mature [27], and many such automated tools exist [20, 21, 26, 36, 18, 25]. This automation is often built into the system so that verification is continually addressed throughout knowledge base construction to ensure a quality final product.

Testing, including validation, is best done throughout the entire development of the knowledge base. Incremental testing can aid in finding inaccuracies or incompleteness early in the development of the system rather than later when corrections can be much more difficult to detect, locate, and correct. In determining the overall validity of a system, it is often beneficial to determine how well human experts do in the problem area and to create reasonable expectations of the systems performance [15]. Typically, expert systems and their knowledge bases' performance can change drastically from initial release to later stages of use. Some systems can be field tested and validated in its early use without harm. In critical applications where lives may be at risk, field testing is not always possible[1]. The expert whose knowledge was modeled should maintain involvement throughout development of the system whenever possible. This can often assist in identifying errors early on in the development cycle that may not have been detected until later stages of validation.

Validating after modifications or enhancements have been implemented is just as important as earlier testing. Testing needs to ensure that the original system was not degraded as well as that the modifications made were correctly implemented. Comparison of previous test case results and their performance after the modifications is an effective way of testing the updated system remains validated in areas both inside and outside of

---

[1]An exception exists in some cases when the system can be ran in conjunction with a human expert.

the modified areas. This type of testing can be particularly important in probabilistic representations, since chains of inference can be unintentionally altered.

2.2 Methods of Verification & Validation

We have discussed the importance of V & V testing, and when this testing is appropriate. Let's now investigate how this testing is done. Brute force testing of all possible cases is impossible in systems with any size. Understanding the ways in which the knowledge base will be used by a problem solver can help determine a set of test cases to see whether important features of the system's problem solving behavior are being exercised. In some systems a false alarm is not as bad as missing an alarm, while in others the inverse is true. For example, in an air traffic control system a missed alarm could result in a catastrophe, where in some other type of preventive system a false alarm could mean a great deal of time and/or money to fix a problem that actually never existed. Either of these occurrences could equally result in abandonment of the system. In testing a system it is also important to verify that the system used the appropriate line of reasoning in deriving its conclusion. Explanations describing how and why the system arrived at its solution can be extremely helpful. Being able to investigate intermediate rules and constraints used for a result can often lead to errors that would possibly have never been detected in normal testing.

Methods of validation testing are numerous. Using a group of experts for a face validation is common. This group of experts together assess the validity of the performance of the system using an agreed upon evaluation range. A group of six experts validated R1/Xcon, an early expert system, reviewing its performance on 50 orders [22]. Predictive validation, or using historic test cases with known results and measures of human expert performance, is also frequently used. The choice of test cases must be handled carefully. The test cases should extensively test the knowledge base. The number of test cases may not be as important as the coverage of the test cases [22]. Test cases that were used throughout development of the system are obviously not good test cases for validation purposes. The system will certainly have been altered to handle these developmental cases.

Test cases should include obvious conditions, subtle conditions, boundary conditions, and even meaningless combinations of valid and invalid data.

Turing tests are also an effective validation technique. Turing tests solely examine human performance, without knowledge of the system being tested. After acquiring enough data, the examination results can be used to validate the performance of the system. Such tests have been successfully used on a number of early knowledge based systems including MYCIN [10]. Sensitivity analysis can be performed by changing input variable values and parameters over some range of interest and observing the effects. This technique is especially useful for dealing with uncertainty measures [10]. All of the V & V techniques mentioned above have their strengths and weaknesses and can be used in combinations to provide more thorough testing.

2.3 V & V of knowledge based systems versus conventional software

There are some difficulties in applying verification and validation to expert systems that are not typically found in other software systems. An appropriate analogy: evaluating an expert system is to evaluating conventional software as grading an essay examination is to grading a true-false examination [11]. Tests of conventional software yield true-false results, while tests of experts systems yield more complex results. There may be more than one acceptable answer, or there may be more than one way of stating the answer.

As the above analogy implies, one of the biggest difference in conventional versus knowledge based systems is the fact that knowledge contained in these systems can be very subjective. The knowledge they contain are often the impressions and thoughts of a human expert. Experts certainly do not always agree with one another. The same problem can often be given to two experts in a particular field with two different but correct solutions returned. Both experts will certainly state his/her solution is optimal. Another common occurrence is that software requirements and specifications are nonexistent, imprecise, or rapidly changing [11]. It is often argued it is more work to write the specifications than it is to write the knowledge base directly. When systems are built by refinement and customer interaction, requirements change rapidly.

As is typical in the emergence of newer technologies, lack of a design standard is certainly found in knowledge based systems. There have been many different techniques developed to represent knowledge based systems. Example representations include rule-based systems, frames, objects, and semantic nets. Each representation has its unique characteristics leading to different methods of V & V. Other related problems encountered include dependency on other components (e.g. knowledge acquisition tools, inference engines), unreliable human expert evaluations, and a lack of modularity.

2.4 Knowledge Acquisition

Key to the development of any knowledge based system is knowledge acquisition (KA). KA can be defined as the process of extracting knowledge from a source, usually in the form of a human expert or experts, into a computer system. Because of problems in the areas of knowledge elicitation and knowledge representation, KA has often been termed the "bottleneck" of knowledge based system development [22]. KA can be the most difficult and critical component in the development of these systems. Knowledge must be transformed many times over before being used in a knowledge based system. First, a human must acquire expertise in some domain through experience and/or study. Next, the knowledge engineer must somehow extract this knowledge from the expert and express it in the internal representation of the knowledge base. This extraction process is far from trivial. It is often done by means of interviewing the expert repeatedly for weeks or even months, depending on the complexity of the system. Other ways of acquiring this knowledge include observation, intuition, induction, and data mining. A thorough investigation of a variety of knowledge elicitation techniques can be found in Cooke [4]. A process called incremental development is often used during this stage of development [10]. A chunk of knowledge is elicited, implemented, and tested. Once this chunk is developed and tested, another chunk is chosen and the process begins again.

Much work is currently being done in creating automated knowledge based systems. These systems extract data while continually checking for inconsistencies and gaps. As mentioned previously, there are commercially many available system shells for catching common errors in syntax and rule misuse. These tools are invaluable as knowledge bases

increase in size. They are also often beneficial in providing some type of understandable display of the knowledge as the system uses it. Completely automated KA systems, where the domain expert interacts directly with a knowledge elicitation tool to build the knowledge base, can assist in avoiding the pitfalls created when a knowledge engineer interviews an expert. These pitfalls include insufficient understanding of the domain, misinterpreted information, and the amount of interview time of both the knowledge engineer and expert(s). The MORE [12] system is an example of an automated KA system that helps refine an existing knowledge base. SALT [16] is a KA tool that identifies weaknesses in its knowledge base and tailors the KA process with the expert to strengthen these areas.

2.5 Knowledge Representation

A knowledge representation scheme suitable for the problem domain is critical to the success of any KBS. The knowledge engineer must weigh the strengths and weaknesses of different representations in an attempt to decide upon the most suitable one for the particular problem domain. Rule-based systems are by far the most commonly used type of representation. This choice is partly due to the natural expression of knowledge by humans as condition/action relationships [10]. These rules are most often in the form of If-Then expressions. Other representations include semantic nets, frames, objects, logic, or a combination of these.

One major consideration when choosing a representation is its handling of uncertainty. Uncertainty is an important and difficult problem in the development of knowledge based systems and is found in most tasks requiring any kind of intelligent behavior. Humans constantly accomplish handling uncertainties, but getting a computer to deal with uncertainty is much more difficult. There are many sources for uncertainty in systems. Data is often missing, unreliable, ambiguous, conflicting, or even just a user's best guess. Given this, uncertainty schemes have been developed to represent these cases. These instances are where mathematics, particularly probability, makes its mark on knowledge based systems. Schemes dealing with uncertainty include Fuzzy logic [38], certainty factors [35], influence diagrams [32], Dempster-Shafer Theory of Evidence [5, 33], Bayesian

probabilities [24], and the representation used in this research - Bayesian Knowledge Bases [30]. Each of these approaches have their unique advantages and disadvantages.

2.6 The Bayesian Knowledge Base representation

BKBs are a new, powerful, and highly flexible knowledge representation [30]. BKBs are closely related to Bayesian networks [24] and in fact subsume them. BKBs, just as Bayesian networks, are strongly based upon probability theory. This foundation allows a framework for enabling inferencing over incomplete knowledge. In contrast to BKBs, Bayesian networks demand for a complete specification of probability distributions can make knowledge acquisition, knowledge base creation, and inferencing quite difficult and cumbersome. When incompleteness exists in Bayesian networks, inferencing is not possible. Even when no incompleteness exists, the computation for computing belief networks through conditional probabilities given some observed evidence is in fact NP-hard [3]. These limitations of Bayesian networks have been overcome through the use of BKBs. BKBs avoid an over-defined system easing maintainability, verification, and validation. They are more powerful from the fact that they are specifically designed for allowing incompleteness [30]. However, when desired conclusions are unable to be drawn from the knowledge base given the appropriate evidence, this incompleteness needs to be corrected through incorporation into the knowledge base.

In the BKB representation, as in Bayesian networks, random variables (RVs) are used to represent objects and/or events in the world. These RVs are then instantiated with state values and are used in combination with one another to model the current state of the world. Inferencing over this knowledge representation then involves computing the joint probabilities of these RVs. This type of inferencing is known as belief revision. Belief updating is also possible and involves finding the probability of any I-node based upon some evidence. Belief revision is more useful in diagnostic domains where the exact probability of any particular element is not as useful as the inclusion of an element [9].

BKBs are built through the combination of instantiation nodes, support nodes, and arcs. An example BKB is shown in figure 2.1. Instantiation nodes, or I-nodes for short, are represented by an oval. An I-node represents one instance of an RV. The arcs represent

2-7

the relationships between these I-nodes. Support nodes, or S-nodes, are represented by smaller rectangles or circles. S-nodes are assigned probabilities that are associated with one or more I-nodes. In figure 2.1, I-node **Clouds = Heavy** is supported by a single S-node with a probability of 0.1500. I-node **Sidewalk = Wet** is supported by the single I-node **Clouds = Heavy** through an S-node probability of 0.8500. In order for the S-node to be active, the supporting I-node, in this case **Clouds = Heavy**, must be active.

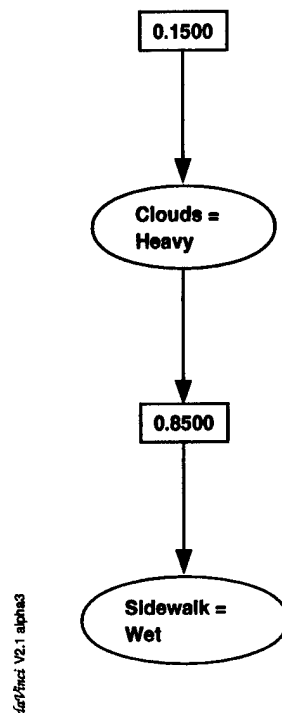

Figure 2.1   **Example BKB.**

Some constraints of this BKB representation include the following:

- All I-nodes must have at least one parent S-node.

- All I-nodes are unique. Different instantiations of the same RV must have distinct values.

- Support conditions for an RV instantiation must be mutually exclusive. Only one S-node may be active at any one time in the support of an I-node.

- Cyclic knowledge is not allowed. A node is not allowed to support itself.

- Only one instance of an RV may be active at any one time. Being concerned with only one particular state of the world at a time mandates this constraint.

- Probabilities from the same RV may not sum to values greater than 1. Probabilistic reasoning demands do not allow this.

For a complete discussion of these constraints see Banks [2].

2.7 Knowledge Base Errors

Imperfect information is ubiquitous - almost all the information that we have about the real world is not certain, complete, or precise [23]. It is critical for the knowledge engineer to understand that these conditions exist during creation of a knowledge base. Three concepts that are essential for V & V of BKBs are inconsistency, incompleteness, and incorrectness [31]. Inconsistency in a BKB is primarily related to probabilistic values. For example, conditional probabilities summing to greater than one. These types of errors are often discovered and corrected within the knowledge acquisition process. When this form of inconsistent knowledge is introduced into the PESKI system, the knowledge engineer is immediately informed through a continuously updated status window. This process assures that knowledge is consistent throughout the entire knowledge building process.

Incorrectness is probably the most difficult of errors to detect and correct, and is certainly the least addressed area of validation. This form of error occurs when a query to the system results in an incorrect solution. Finding the location of the error can be difficult, and correcting it even harder. This aspect of validation will continue to be addressed through a variety of approaches such as sensitivity analysis and neural network reinforcement learning techniques [31, 29].

Incompleteness exists when a set of input values is passed to the system and fails to arrive at a conclusion. This type of omission can be very difficult to detect and locate as well. Knowledge base incompleteness can be both intentional, particularly in the case of

BKBs, or unintentional as in an oversight. Incompleteness can come from several different sources. Human error is often the major source for this type of error. Experts often have difficulties in conveying complete heuristic knowledge to the knowledge engineer. This lack of information often leads to incompleteness in the knowledge base. Often information is missing during development of the knowledge base and is left out for future modifications. Other types of knowledge are yet to be discovered, particularly in some areas such as medicine, where new types of drugs and medications are constantly under development. For these and other reasons, the ability to handle incompleteness is critical in the validation of these systems. This ability is even more critical in a representation like BKBs, in which the ability to incorporate incompleteness is an advantage and a normally desired quality of the representation.

Incompleteness can occur in several different ways:

- Missing links - Relationships between I-nodes are missing.

- Missing RVs

- Missing states

Each of these are addressed in this research and are further discussed in the following chapter.

The origin[2] of incompleteness in a BKB is the direct dependency region[3] of an evidence item from a test case. This direct dependency region must be modified through the addition of a link or links to the corresponding answer item or it's direct dependency region in the BKB. This addition of a link must be done in a manner that places the answer item in the direct dependency region of the evidence. Only then is the incompleteness dissipated. Without some type of automatic correction, the knowledge engineer must be presented the information contained in the BKB in a manner suitable for the addition of this incompleteness. The methodology developed through this research graphically presents the BKB in a manner suitable for a knowledge engineer to locate this area of

---

[2]Origin in the sense that this is where you will begin the search for the necessary location of the incomplete link.

[3]Direct dependency regions are defined and discussed in detail in section 3.2.2.

incompleteness. The incompleteness link can then be added to the BKB for future infer-encing. This link is added by the tool in a manner that forces maintaining the rules of the BKB representation. The next chapter discusses further these ideas and the methodology developed through this research.

III. Methodology

This chapter describes a methodology to handle incompleteness caused by missing links in a Bayesian knowledge base (BKB). The incompleteness links are identified and located in the BKB using test cases. As discussed in the previous chapter, test cases are a commonly used method for validating knowledge bases. When test cases lack a direct dependency connection (formally defined in section 3.2.2) from the evidence and answer items, the knowledge engineer previously had only the option to begin the data mining utility, or return to the normal knowledge acquisition mode to correct the incompleteness. This chapter will introduce another method of handling incompleteness through a graphical approach. This graphical presentation of the BKB gives the knowledge engineer a means of locating and correcting the incompleteness found in the test case. Some other desired traits of this methodology include the following:

- Maintain the structure of the previous information contained in the BKB. Correcting the incompleteness while at the same time removing other necessary information contained in the knowledge base is obviously an ineffective way of performing validation. An assumption is made that information not addressed by the current test case in the BKB is correct. Any structural changes the methodology permits are contained in the evidence/answers direct dependency regions.

- Not allow the knowledge engineer to input information into the BKB that violates the knowledge representation rules (e.g. circularity, mutual exclusion, etc) [2]. The ability to allow only solutions that do not violate BKB knowledge representation constraints should be presented to the user.

- Inherent with any large knowledge base is the problem of finding where the incompleteness exists in order to fix the problem. The tool should avoid presenting too much information that overwhelms the knowledge engineer, particularly in larger knowledge bases.

- Similar to the first goal, correcting one area of incompleteness will not undo a previously made correction.

3.1 PESKI Validation

This section defines some important terms, and discusses the methodology used to handle validation in the PESKI system.

3.1.1 Test Cases.
Test-cases are formally defined in the following definition from Lyle [14].

Definition 1 *Let A be the set of all random variable instances specific to the test-case as answers, and $A \neq \emptyset$.*
Let E be the set of all random variable instances specific to the test-case as evidence, and $E \neq \emptyset$.
Let random variable(A_i) be the random variable of the ith element of A.
Let random variable(E_j) be the random variable of the jth element of E.

- *For all elements A_i in A, there does not exist an element A_j in A, $i \neq j$, such that the random variable(A_i) = random variable(A_j).*

- *For all elements E_i in E, there does not exist an element E_j in E, $i \neq j$, such that the random variable(E_i) = random variable(E_j).*

- *For all elements A_i in A, there does not exist an element E_j in E such that the random variable(A_i) = random variable(E_j).*

- *For all elements E_i in E, there does not exist an element A_j in A such that the random variable(E_i) = random variable(A_j).*

As mentioned previously, test cases are the basis for validation in the PESKI system. These test cases are submitted to the system and the results are compared to expected results. The test cases are used to validate the BKB through a number of steps using the PESKI system. Figure 3.1 depicts the flow of a test case through PESKI. The PESKI system validation tools place some assumptions upon the supplied test cases. The test cases used in knowledge base validation are constrained in that they are assumed correct in their entirety. The knowledge engineer's burden is to ensure that each test case is completely valid. In addition, it is important to understand that each evidence item is

directly related to each answer item, and vice versa. If this is not so, the non-contributing evidence or answer item should not be a part of the "valid" test case. Since all answers are in each evidence item's dependency region, the intersection of all the evidence dependency regions should be non-empty and contain, at a minimum, all answer instances [14]

PESKI internal validity tests also ensure there are at least one evidence and one answer item in the test case, and that the evidence/answer(s) are currently contained in the knowledge base. Incompleteness existing from missing random variables and/or states in the knowledge base must be handled through the regular knowledge acquisition process or through data-mining information queries. The PESKI data mining and knowledge acquisition tools allow the addition of the necessary random variables and their instantiations to the BKB for future validation efforts. The verification and validation (V & V) user interface only allows for RVs and states already existing in the currently loaded BKB for selection as either an evidence or answer node. If the test case is invalid, resubmission of a valid test case is required. If the test case is valid, it's then checked to ensure enough information is contained in the knowledge base so that inferencing can occur. This check uses the concept of direct dependency regions which are described in the following section.

3.1.2 Direct Dependency Regions. Direct dependency regions are the key to validation efforts in the PESKI environment and are formally defined below. RV instances directly dependent upon one another are connected by a sequence of parent or child relationships. Therefore, I-node A is directly dependent upon I-node B if there is a sequence of parent nodes, or a sequence of child nodes, between A and B that connect the two nodes. Figure 3.2 shows the direct dependency region for the evidence item **D = 1**. Formally from Lyle [14],

Definition 2 *A random variable instance A is* **directly dependent** *on a random variable instance B, if and only if there exists a sequence of n random variable instances*

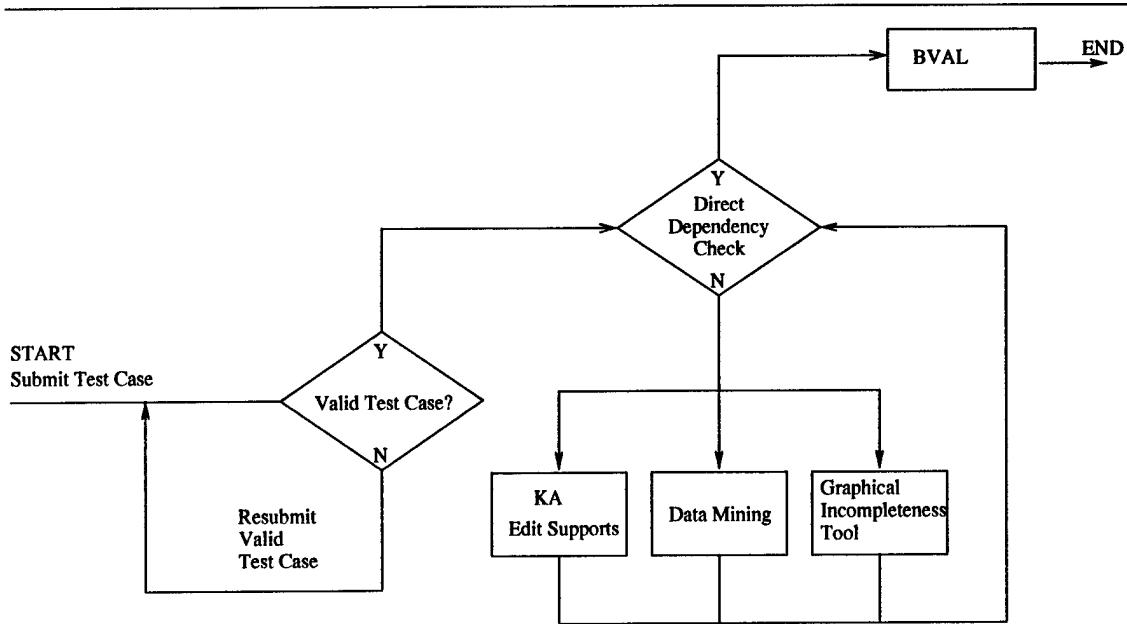$\{A, X_2, ..., X_{n-1}, B\}$, *where n is positive integer, and*

Figure 3.1   **Test Case Flow.**   Test case flow through the PESKI system

1. *Each element S_i in the sequence of random variable instances is an element of a support condition of S_{i-1}, for all i, $2 \leq i \leq n$,*

or

2. *Each element S_i in the sequence of random variable instances is an element of a support condition of S_{i+1}, for all i, $1 \leq i \leq n-1$.*

After validating a test case, the next step is to check for direct dependency region connections between the evidence and answer items. If the evidence and answer(s) are found to be both contained in the same direct dependency region, in the case that no incompleteness exists, the BKB is then passed to a tool for probabilistic validation. This tool is further described in section 3.2.3.

If the direct dependency region check fails, incompleteness exists in the test case and needs to be corrected in the knowledge base structure. The knowledge engineer then has
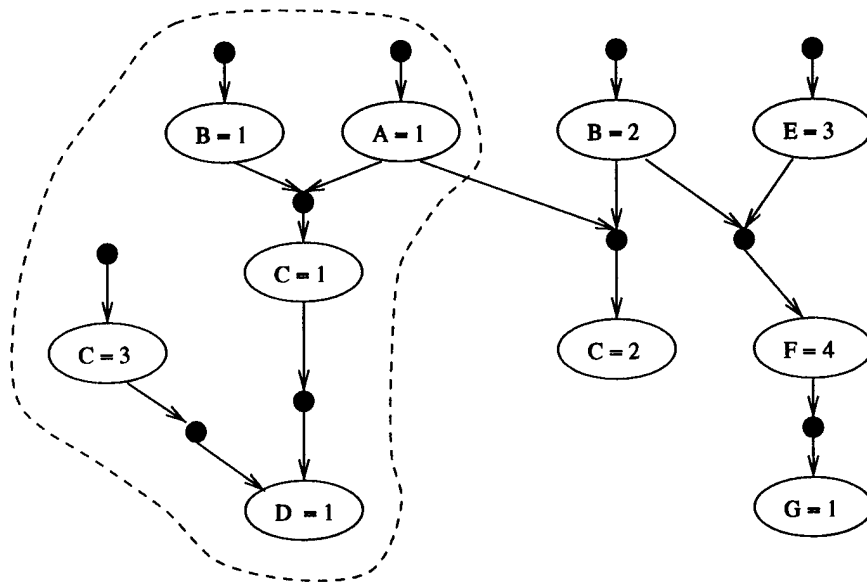
Figure 3.2   **Direct Dependency Region.**   Region within dotted line is the direct dependency region of item **D = 1**.

three options for correcting this incompleteness. The PESKI data mining tool can be used if the knowledge engineer feels electronic information may be available to the tool that will correct the BKB incompleteness. This information can be in the form of data files, text-based web pages, on-line encyclopedias, and others. If the data mining process is unable to solve the incompleteness, the graphical incompleteness tool built within this research can be utilized. If the incompleteness requires complex modifications to the knowledge base and data mining is not able to solve the problem, the normal knowledge acquisition mode in PESKI should be used. Complex modifications can include the addition of missing I-nodes and/or S-nodes, multiply connected incompleteness links passing through nodes outside of the direct dependency regions of the test cases' evidence and answer(s), the correction of any incorrectness located, or any necessary deletions in the BKB.

3.1.3 BVAL.   When either the identified incompleteness from the test case is corrected or no test case incompleteness exists in the BKB, the BKB is passed to BVAL for further validation testing [9]. BVAL is a tool used for validating probabilistic BKBs.

It incorporates validation through rule-based and neural network techniques to provide some automatic correction of the probabilistic knowledge in the BKB. BVAL incorporates upscaling and downscaling of the probabilities of S-nodes as necessary to ensure the test case is valid in the BKB. A limitation of BVAL is its handling of incompleteness. When BVAL encounters incompleteness, it will either make assumptions about the test case or remove test case information (either evidence or answer items). This can result in invalid test cases and incorrect conclusions by the system. This research identifies incompleteness before BVAL is invoked, so that when BVAL is executed the incompleteness no longer exists. Further details concerning BVAL can be found in Gleason [9].

3.2 Graphical Incompleteness Handling

Although consideration of a methodology to present possible solutions through a textual interface was considered, the combinatorial explosion of possibilities as the size of the BKB increased made this approach infeasible. Therefore, a graphical solution for the incompleteness handling was pursued. A visual interactive approach allows the knowledge engineer to examine the knowledge base for completeness as well as accuracy. The knowledge engineer interacts with the system and actively influences the solution, thereby not forcing the generation of possible solutions from which the user must select a preferred choice. This type of correction also avoids any non-sensical modifications and/or assumptions that an automatic validation tool may make. The ability to view portions of the knowledge base where problems are occurring can provide a great deal more information than a text-based method would provide. Graphical depictions of the BKB were already implemented in the PESKI system for use with other tools, namely the knowledge acquisition phase. The incompleteness methodology extends the graphical presentation of the BKB into a display for allowing incompleteness correction in an intuitive fashion.

Ideally, incompleteness would be handled through some type of automatic process that would ensure correction. However, automatically performing this correction requires some "sense" and awareness of the knowledge that is missing. If this "sensing" was possible, then the incompleteness should already have been corrected by the system and not ever allowed to exist in the knowledge base. Previous work by Lyle [14] attempted this

automated form of correction. These manipulations involved examining the nearby rela-
tionships of the evidence and answer nodes and attempting to correct the incompleteness
discovered. Lyle states one limitation of this approach is indeed the idea of "choice." With
any automated choice, there is the possibility that invalid information will result. Assump-
tions and modifications that were made during Lyle's work proved to be invalid in other
domains.

The most reliable source for incomplete knowledge leads back to the expert or the
knowledge engineer. Data mining provides a mechanism for automatic incorporation of
knowledge into a knowledge base; however, this incorporation can only occur if the knowl-
edge is contained in the data that is to be mined. Certainly this electronic form of informa-
tion will not always be available, particularly when the knowledge base was created from
this information. Completely automatic changes may also be unwelcome to a knowledge
engineer who has been creating a knowledge base for what is often a very lengthy period of
time, without, at a minimum, the approval of these changes. For these reasons, the knowl-
edge engineer should be responsible for the handling or approval of any modifications of
the knowledge base.

3.2.1 Methodology. We have defined incompleteness in three ways.

- Missing links

- Missing RVs

- Missing states

As mentioned previously, incompleteness caused by missing RVs or states can be cor-
rected through data mining or through the normal knowledge acquisition modes of PESKI.
Of concern in the remainder of this research is unintentional incompleteness caused by miss-
ing links or relationships between I-nodes. After the test case direct dependency region
check has discovered this type of incompleteness, the methodology begins by presenting
the BKB in a graphical format that allows for the location of the incompleteness. This is
further aided by distinctly displaying the direct dependency regions of the evidence and
answer items from the incomplete test case in the BKB. The ability to traverse anywhere

in the BKB is also allowed to assist the knowledge engineer by allowing him/her to search for pertinent information. The incompleteness link should be allowed irregardless of the location of the nodes within the BKB, as long as the BKB representation constraints are not violated. The addition of a link can cause constraints to be violated if care is not taken to avoid these situations. Some links between nodes cannot be allowed at all, while others must be specially handled.

This methodology contains two modes of incompleteness correction - an add mode as well as an insert mode. These modes allow for correction of incompleteness in several forms. The methodology embodied in these modes, as well as examples of their usage will be discussed further.

After locating the source node(s) of incompleteness, the knowledge engineer is allowed to select an extend[1] I-node for correction of the incompleteness. The extend I-node must be either the evidence or answer node or a direct dependency descendant of either the evidence or answer node. The location of the target S-node is dependent upon the currently active mode. In add mode, the target node must be a direct dependency ancestor of the corresponding evidence or answer node. In insert mode, the target node may be any node, either descendant or ancestor, in the direct dependency region of the corresponding evidence or answer node. These restrictions ensure that the directed link will be placed in the correct cause/effect direction and will also resolve the current incompleteness problem. Selection of the extend and target nodes in any other manner will not correct the incompleteness due to the direct dependency relationships between the nodes.

After extending the I-node, these allowed target S-nodes are clearly identified to the knowledge engineer. These allowed target S-nodes will guarantee the correction of the incompleteness and avoid any BKB constraint violations. Reasons for a non-colored S-node in each of the two separate modes follow:

In add mode:

---

[1]The extend I-node can be considered as a causal node, while the targeted node is the effected node. This terminology previously existed in the graphical knowledge acquisition mode of PESKI.

1. The S-node is in the child segment of the target's direct dependency region and would not solve the incompleteness problem if targeted.

2. The S-node is not in the direct dependency region of the current test cases evidence or answer's item (depending upon which was extended), and will not provide a solution to the current incompleteness problem.

3. The S-node is a parent of an I-node that is an instantiation of the same RV and cannot be dependent upon one another due to Constraint 6 of the BKB representation rules from Banks [2].

   In insert mode:

1. The S-node has a child I-node or parent I-node which is an instantiation of the same random variable and cannot be connected to one another due to Constraint 6 from Banks [2].

2. The S-node is not in the direct dependency region of the current test cases evidence or answer's item (depending upon which was extended), and will not provide a solution to the currently focused incompleteness problem.

3.2.1.1 Add Mode. The add mode allows for the addition of a link between an extended I-node and a targeted S-node. The add mode never removes any links, and therefore avoids creating any added incompleteness. This addition alone also ensures that correcting one area of incompleteness does not invalidate any previously made corrections. After the location of the needed incompleteness link has been found and the selection of the extend I-node and target S-node have been selected, with the exception of one case, the link may be simply added. If the selected target S-node has as a parent a different instantiation of the same RV as the extended I-node, the link is not able to be simply added. In this case, a new S-node must be created before the addition of any links. Probabilities of any affected or added S-nodes must then be modified by the knowledge engineer. At no time are probabilities modified automatically through this graphical incompleteness methodology.

The following methodology is introduced for the addition of a link:

- If the targeted S-node has a parent with the same RV:

    1. Create a new S-node.

    2. Link the extend I-node and the target I-node through the new S-node with the addition of links.

    3. Link the remaining parents of the original targeted S-node, other than the common RV I-node, to this new S-node.

- Else,

    1. add the new link from the extended I-node to the targeted S-node.

- The probability of any affected S-node, whether new or structurally modified, can be adjusted through the interface menu.

    3.2.1.2 Insert Mode. The insert mode allows for the insertion of a selected I-node into the region containing a targeted S-node. The location of this target S-node becomes a new location for the selected I-node, while maintaining any parent and child relationships the selected I-node previously contained. The target S-node becomes a child of the extended I-node. Again, probabilities of any affected S-nodes must then be modified by the knowledge engineer. The following methodology is introduced for the insertion of an I-node:

- Create a new S-node.

- If only a single S-node exists above the extended I-node with no further parents above the S-node, delete this parent S-node. This is to prevent the mutual exclusion conflict of Constraint 4 from Banks [2].

- Delete the links, if any, from the targeted S-node to its parents.

- Link the parents of the targeted S-node, if any, to the new S-node.

- Link the new S-node to the extend I-node.

- Link the extend I-node to the target S-node.

- The probability of any affected S-nodes, whether new or structurally modified, can be adjusted through the interface menu.

3.3 Summary

Summarizing this chapter, a high level methodology was introduced for PESKI validation. This methodology involves test case validation using the concept of direct dependency regions to locate incompleteness in a BKB. The different forms of incompleteness and how each of these may be addressed through the existing data mining utility and knowledge acquisition modes of PESKI, as well as the graphical incompleteness handling methodology introduced within this chapter, are discussed. This methodology uses a graphical representation of the BKB to assist the knowledge engineer in locating and correcting the area of incompleteness. Two modes, an add mode and insert mode, are introduced for the addition of this incompleteness link. The next chapter will introduce the graphical incompleteness tool which embodies the methodology introduced in this chapter.

IV. Graphical Incompleteness Tool

This chapter introduces the tool developed through this research and based upon the methodology introduced in the previous chapter. The incompleteness tool first begins by presenting a graphical depiction of the BKB, which contains the currently focused evidence and answer items, as well as a varying number of direct dependency related nodes. The evidence and answer direct dependency regions are presented in unique colors from the remainder of the BKB for easy identification (See figure 4.1). The evidence and answer items themselves are also easily identified through the distinct rhombus shape of the I-node itself. The evidence in this case is the node $b = 1$, while the answer is the node containing $f = 1$. In this particular example, the entire BKB is displayed to the user. It is important to note that the evidence and answer items may share areas in their direct dependency regions; however, incompleteness may still exist. These shared areas of direct dependency region are displayed in a unique color as well. An example of shared direct dependency regions is shown in figure 4.2.

When the knowledge base is small enough (which is most likely rare), one can examine all relationships in order to ensure completeness and accuracy. When the knowledge base is large, care has to be taken that an overload of information does not occur. The number of direct dependency region parent and child nodes, or levels[1] , first presented to the user of the tool is dependent upon the size, particularly the width, of the first level of the direct dependency region. In order to avoid an overload of information to the user, the tool will present a number of levels that, at a maximum, will contain three levels up and down the direct dependency region for both the evidence and answer nodes. The user has the ability, as needed, to traverse up and down the direct dependency regions to locate any particular nodes of interest or nodes necessary for correction of the incompleteness. This action is accomplished by selecting a node in the current display region, which then displays any parent or children nodes of the selected node, if not already displayed. These new nodes will be displayed to the user in their appropriate shapes and colors.

---

[1]A level can be considered one generation of parents and children from the focused I-node, therefore two levels would be two generations of both parents and children.
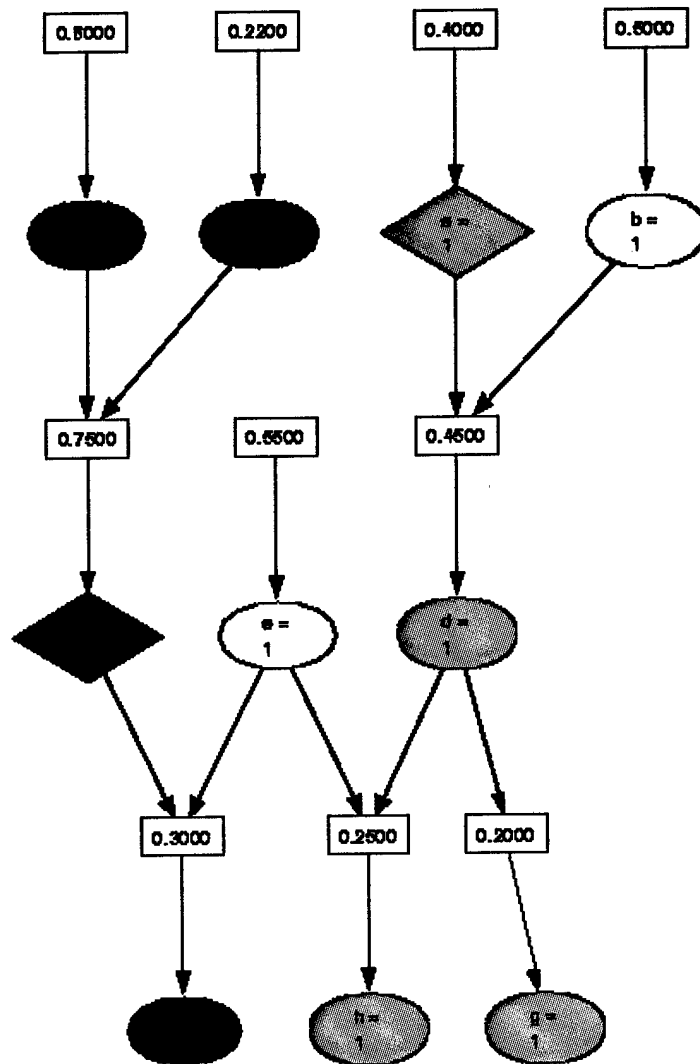
Figure 4.1 **Direct Dependency Regions.** The direct dependency region of the evidence **a = 1** is shown in orange, the answer **f = 1** is shown in green.
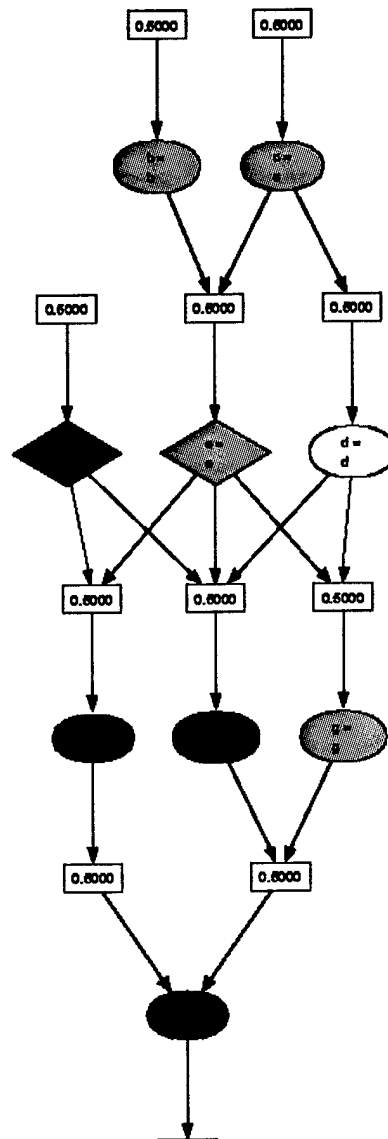
Figure 4.2   **Overlapping Direct Dependency Regions.**   The overlapping direct dependency region of the evidence **e** = **e** and answer **f** = **f** is shown in blue.

4-3

The user also has the ability through the user interface to display any I-node whether the item is contained in a direct dependency region or not. This "goto" feature allows for locating other instantiations of RVs in the BKB that can sometimes be used to help more quickly identify where missing links of incompleteness should exist.

The user, in order to correct the incompleteness, must select areas within the direct dependency regions of the evidence and answer to be connected to one another. The connections allowed by the tool maintain the allowable structure of the BKB representation by not permitting any violations of established constraints. It's important to realize test cases can be presented with the evidence item(s) located anywhere in the BKB, even at the bottom leaves of the BKB structure. The same can be stated for the answer item(s) in a test case. The tool allows for the ability to handle the incompleteness correction in either cause/effect direction.

It may be necessary to run the tool multiple times for any given test case, dependent upon the number of evidence and answer items, as well as the quantity of incompleteness present in the test case. This is partly because of the fact that fixing one link of incompleteness may or may not fix all remaining incompleteness in a test case. The tool presents one incomplete occurrence at a time, with the relevant evidence item and answer item. Only after correcting the current incomplete occurrence, does the tool present any remaining incompleteness occurrences that may still exist in the test case. Therefore, the tool may execute a maximum number of times equal to the cardinality of the evidence items times the cardinality of the answer items in the test case.

The user must first select an extend I-node as well as a target S-node for incorporation of a new link into the BKB. As mentioned in the previous chapter, the selected I-node must be the evidence or answer, or a child of either. This ensures the connection will be placed in the proper direction. After selection of an extend I-node, the tool displays highlighted S-nodes that may be selected for addition of a link while at the same time correcting the incompleteness between the evidence and answer items. The allowable S-nodes are displayed in red for identification purposes and are dependent upon which mode the user is currently in. Figure 4.3 shows an extended I-node in add mode, in this case $b = 1$, and the selectable target S-nodes, which are displayed in red. At any time, the user is allowed

to change modes through the menu interface, which will unselect all I-nodes and reset any colored S-nodes currently in the BKB. The following sections provide further information and demonstrates the methods developed. Appendix B also discusses a measure of BKB incompleteness and the graphical incompleteness corrections' effects relative to a complete Bayesian network.

4.1 Add Mode

The add mode allows for the addition of a link between an extended I-node and a targeted S-node. The add mode does not remove any links, and therefore, does not create any additional unintentional incompleteness. There is only an addition of information to the BKB. This addition alone also ensures that correcting one area of incompleteness will not invalidate a previously made correction. There are two separate cases that can occur in this mode:

Case1: This case occurs when a simple addition of a link is required, and there are no common RVs found between the extend I-node and the surrounding I-nodes of the target S-node. Figures 4.3 through 4.5 demonstrate case 1 sequentially. In this test case, $b = 1$ was presented as evidence and $f = 1$ as the answer. The I-node $b = 1$ was selected as the extend node. Notice the S-node above the I-node $A = 2$ is not allowed to be targeted because it's a different instantiation of the same RV and therefore may not be linked to the extended I-node $A = 1$. The S-node with a value of 0.7500 is displayed in red as an allowable target S-node. After targeting this S-node as is shown in figure 4.4, and next extending, the resulting BKB is displayed as in figure 4.5. The probabilities of any affected S-nodes may now be adjusted through the interface menu. It can be seen that the evidence and answer are now contained within the same direct dependency region. This BKB can now be passed on to BVAL for probabilistic validation.
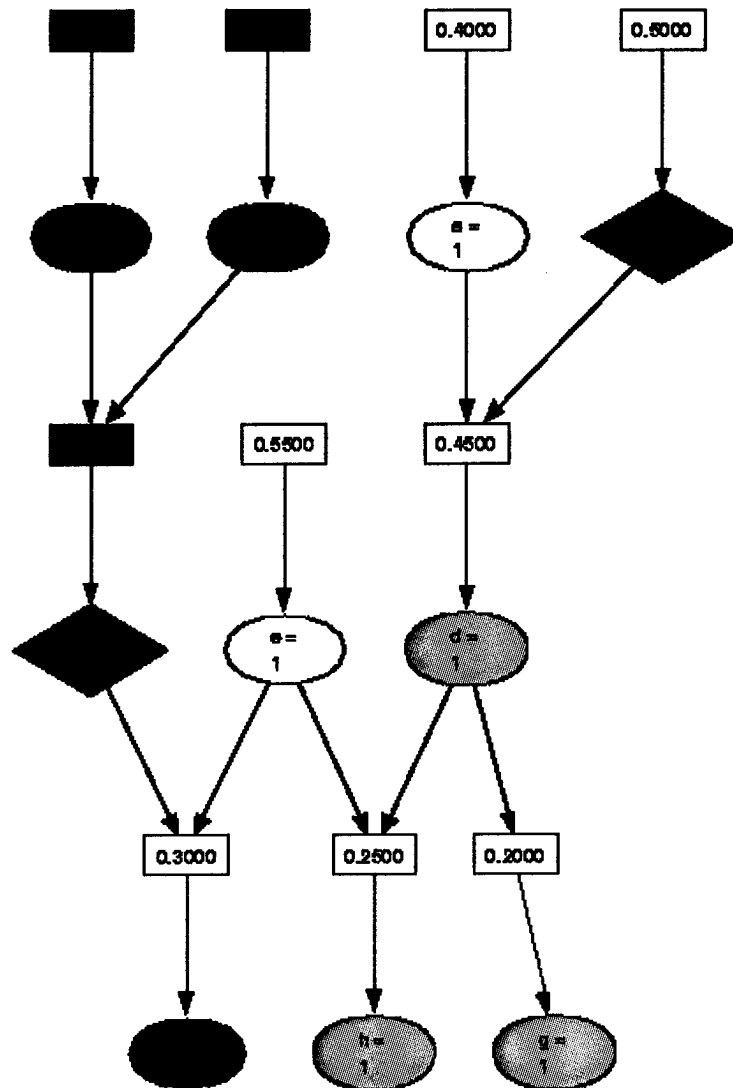
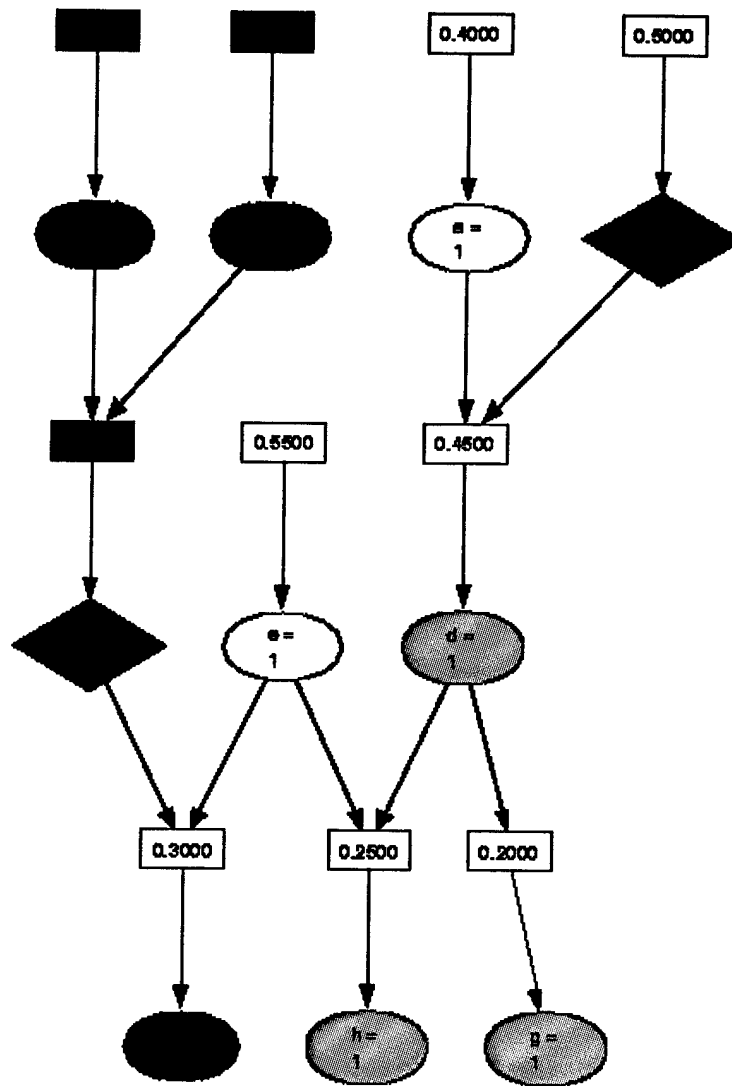Figure 4.3 **Extended I-node in add mode.** Allowed target S-nodes are shown in red.

Figure 4.4 **Targeting S-node in Add mode.** The S-node with value of 0.7500 is targeted for incompleteness correction
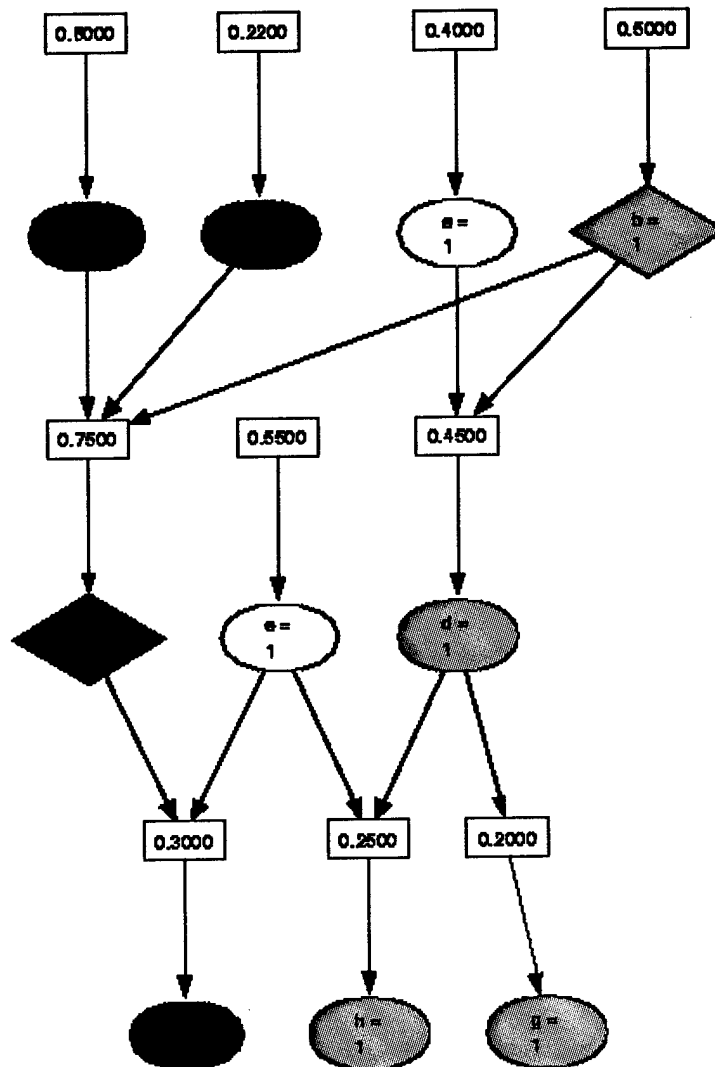
Figure 4.5   **Resulting BKB.** Add mode - Case 1

Case2: This case occurs when an I-node is extended that targets an S-node that has as a parent a different instantiation of the same RV. These nodes cannot be directly linked to one another due to constraint 6 of the BKB representation from Banks [2]. When this situation occurs, the tool must create a new S-node for extend purposes. Figure 4.6 through figure 4.7 demonstrates case 2 occurring sequentially. In this test case, **a = 1** was presented as evidence and **f = 1** as the answer. The I-node **a = 1** was then selected as the extend node. It can be seen that the I-node **f = 1** has as a parent the I-node **a = 2**. The S-node with a value of 0.7500 was highlighted in red as an allowable target S-node. The S-node has also changed form from a rectangle to a circle. This is a signal to the user that if this node is chosen as the target S-node, a new S-node will be created when extended. This is shown in figure 4.6. After targeting the node as is shown in figure 4.7, and next extending, the resulting BKB is displayed as in figure 4.8. The probabilities of the S-nodes, whether new or structurally modified in any way, can then be adjusted through the interface menu. Probabilities of S-nodes are not automatically changed at any time during the usage of the tool.
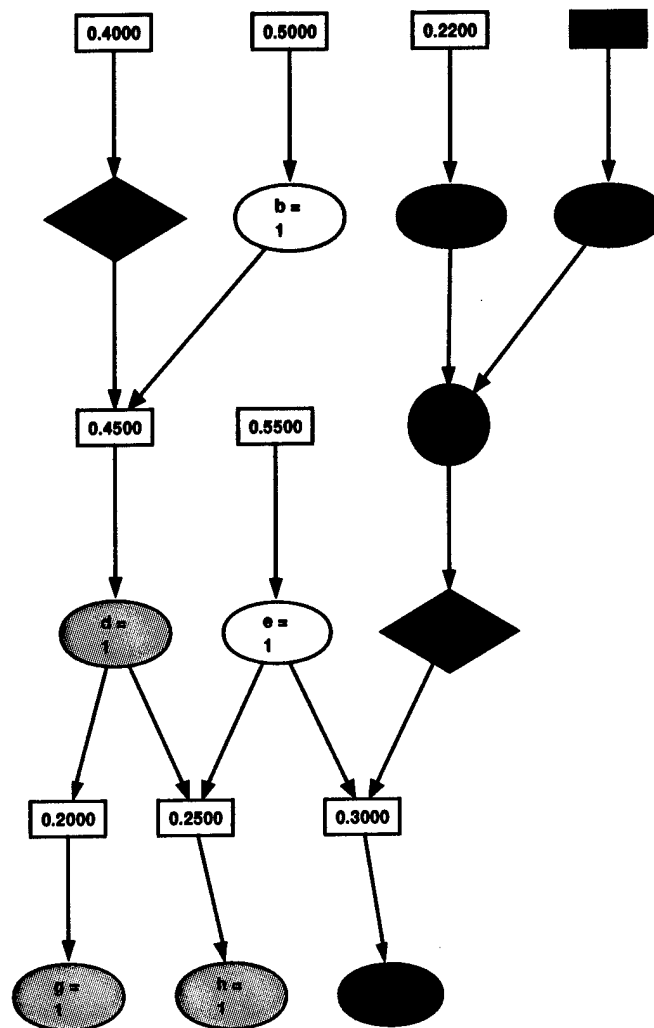
Figure 4.6   **Extending I-node in Add mode - Case 2.**   The I-node **a = 1** is chosen for extending.
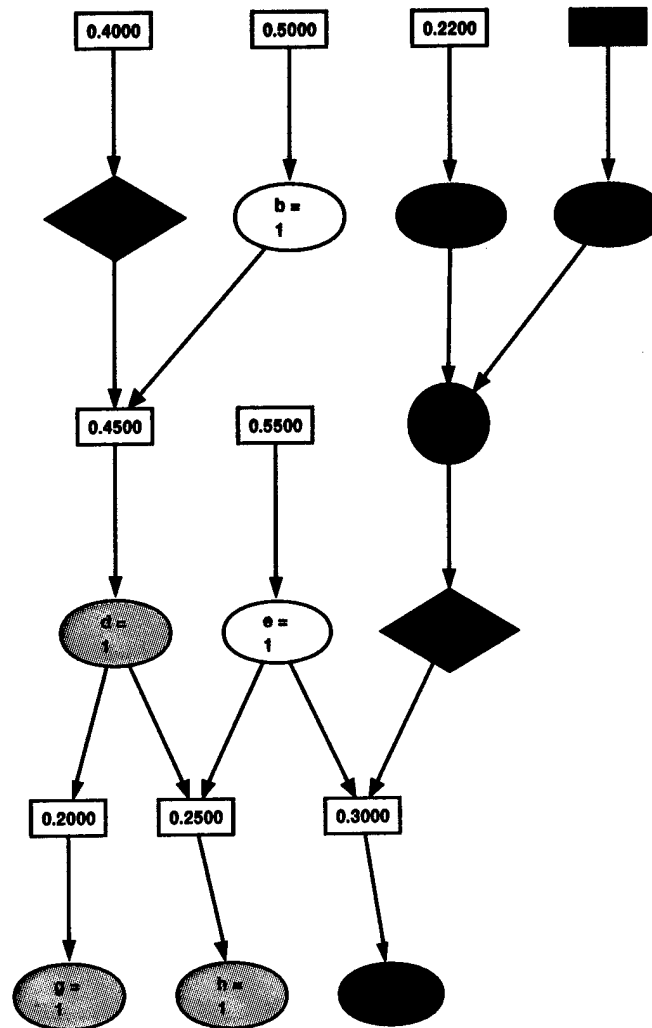
Figure 4.7   **Targeting S-node in Add mode - Case 2.**   The S-node with value of 0.7500 is targeted.
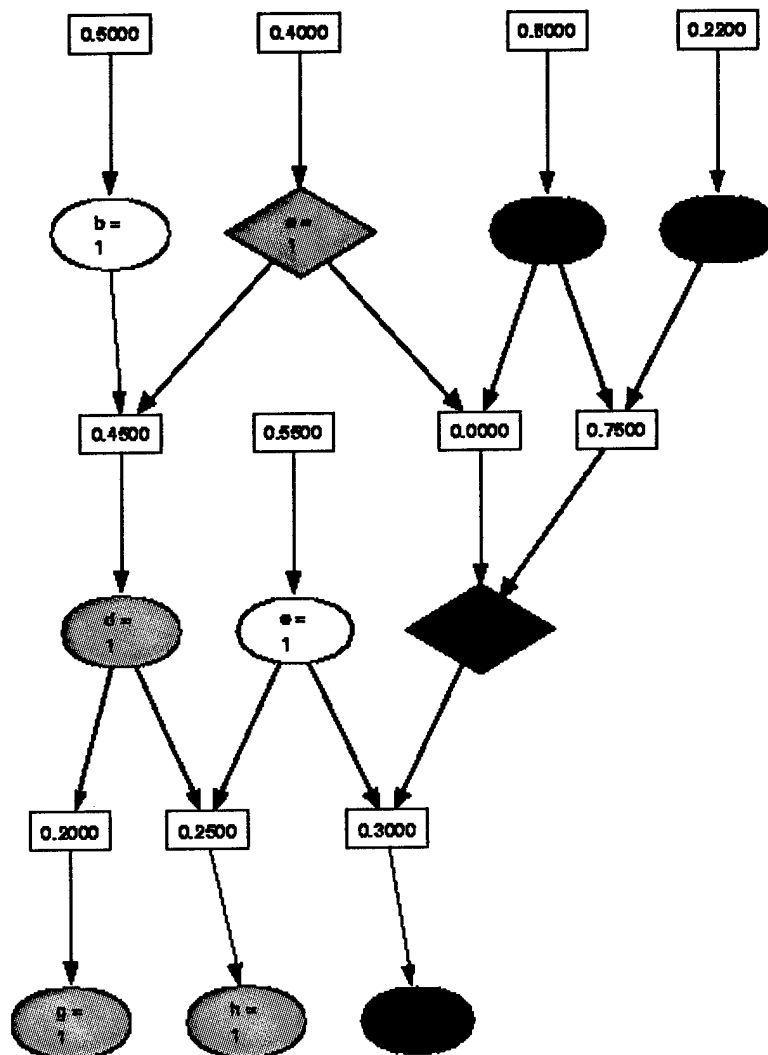
Figure 4.8   **Resulting BKB.** Add mode - Case 2

4.2 Insert Mode

The insert mode allows for the insertion of a selected I-node into the region containing a targeted S-node. The location of the S-node selected becomes a new location of the selected I-node, while at the same time keeping any parent and child connections the selected I-node previously possessed. After selecting an extend I-node, the allowed S-nodes are once again highlighted in red as shown in figure 4.9. This is quite similar to the add mode, except it's important to realize that both the parent S-nodes as well as the child S-nodes are possible candidates for target selection now due to the insertion of the extended node into the direct dependency region of the applicable answer/evidence item. Unlike in add mode, the insertion can correct the incompleteness in either of these parent or child regions. In this example, the test case presented $b = 1$ as evidence and $f = 1$ as the answer. The I-node $b = 1$ was selected as the extend node. The S-node with a value of 0.7500 is displayed in red as an allowable target S-node. After targeting this node as is shown in figure 4.10, and next extending, the resulting BKB is displayed as in figure 4.11.
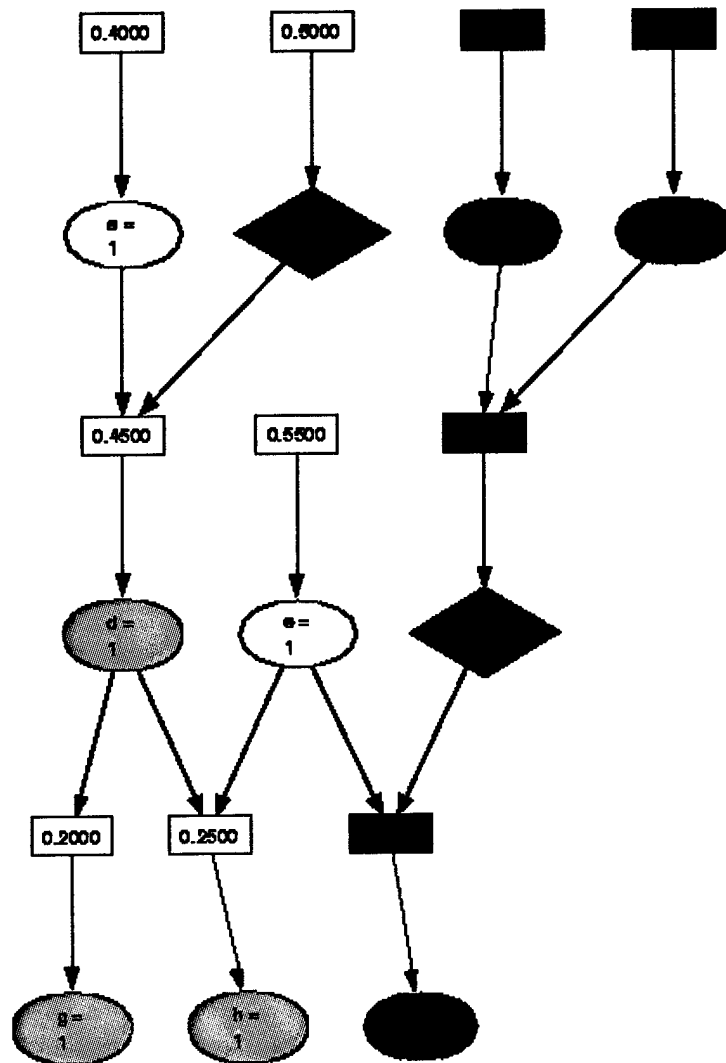
Figure 4.9   **Extending I-node in Insert mode.**   The I-node **b = 1** is chosen for extending.
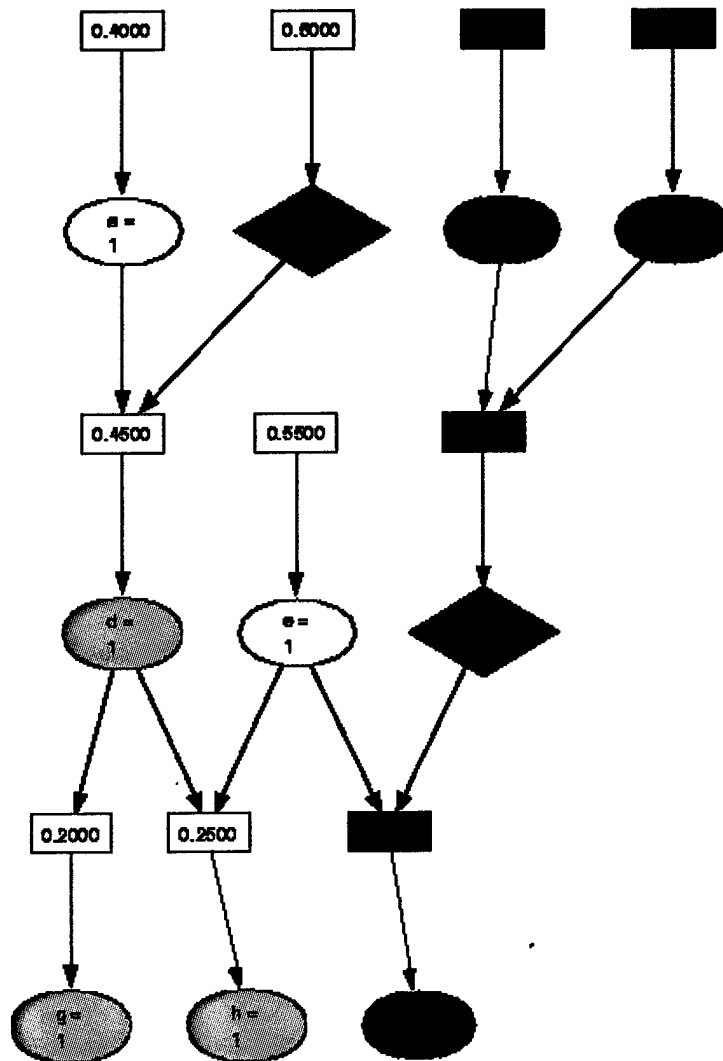
Figure 4.10   **Targeting S-node in Insert mode.**   The S-node with a value of 0.7500 is targeted.
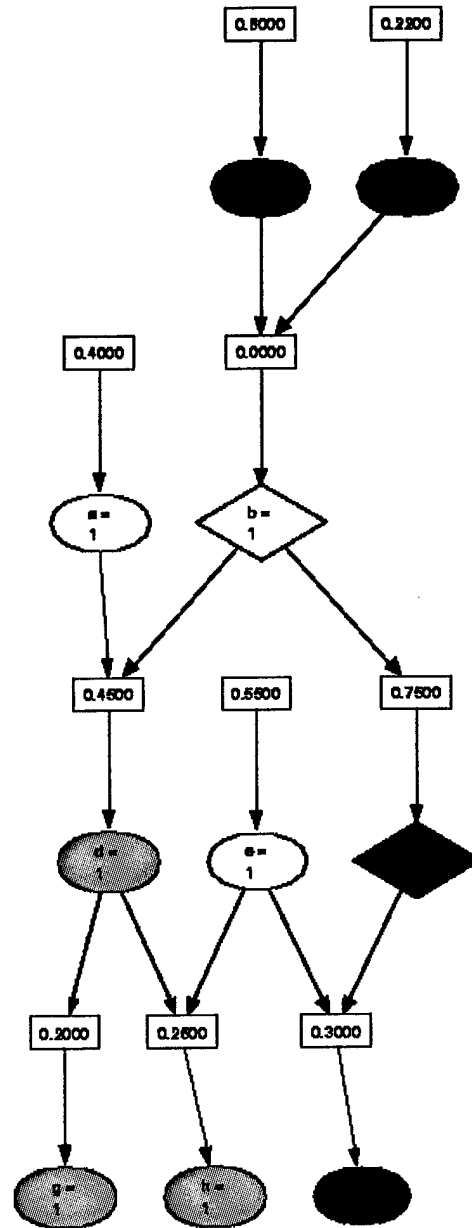
Figure 4.11   **Resulting BKB.** Insert mode

It's important to note mutual exclusion can possibly occur from the changes made in insert mode. Mutual exclusion occurs when the support conditions of an RV state are not uniquely distinguishable from one another. Therefore only one of its supports are allowed to be active at any one moment in time. When a mutual exclusion violation is present in the BKB, the knowledge engineer is informed through the status window. This problem can then be corrected in the PESKI knowledge acquisition tools or the data mining tool. Mutual exclusion is allowed in this mode to give the knowledge engineer freedom to make the needed changes, even if mutual exclusion occurs sometime during these changes.

V. Results

The previous chapter presented the graphical incompleteness tool used for resolving test case incompleteness. This chapter presents some concrete examples of the incompleteness corrections presented in Chapter 3.

5.1 Add Mode - Case 1

The following test case is presented to PESKI:

`Evidence is: Sprinkler = On`

`Answers are: Sidewalk = Wet`

The evidence and answer are not directly dependent upon one another due to incompleteness in the BKB as shown in figure 5.1. In this case **Sprinkler = On** and **Sidewalk = Wet** logically should be connected to one another. These I-nodes cannot be simply linked to one another through an additional S-node due to the mutual exclusion restraint. If **Sprinkler = On** was linked directly to **Sidewalk = Wet**, two I-nodes with their respective S-nodes could be active at the same time. Therefore, the graphical incompleteness tool ensures that the connection is linked to an already existing S-node. The I-node **Sprinkler = On** is extended and the S-node with a value of 0.75 is targeted as shown in figure 5.2. The resulting BKB is shown in figure 5.3. From these results, the evidence and answer items are now contained within each others direct dependency regions, and the incompleteness is resolved.
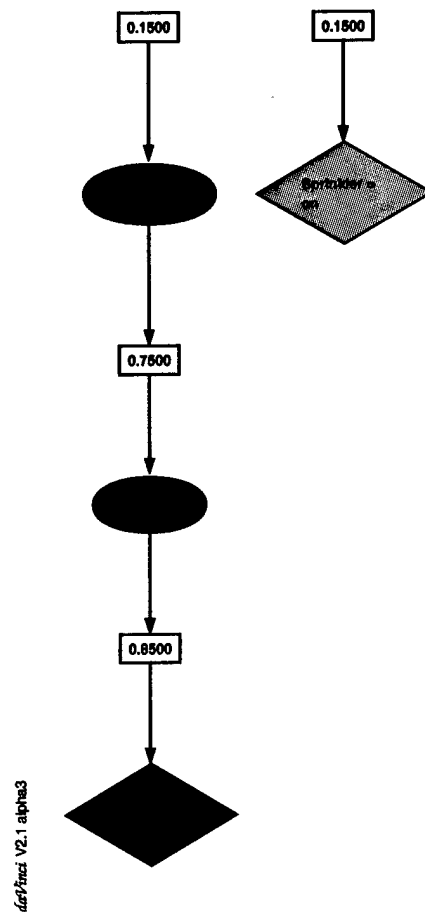
0.1500   0.1500

Sprinkler = on

0.7500

0.8500

daVinci V2.1 alpha3

Figure 5.1   **Add Mode Case 1 .**   Evidence: Sprinkler = On, Answer: Sidewalk = Wet
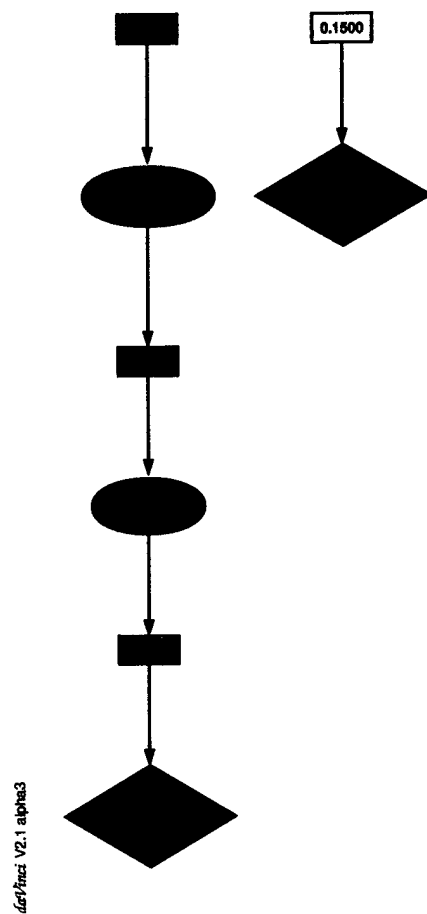
5-2

Figure 5.2   **Add Mode Case 1 - Extended I-node and Targeted S-node.**
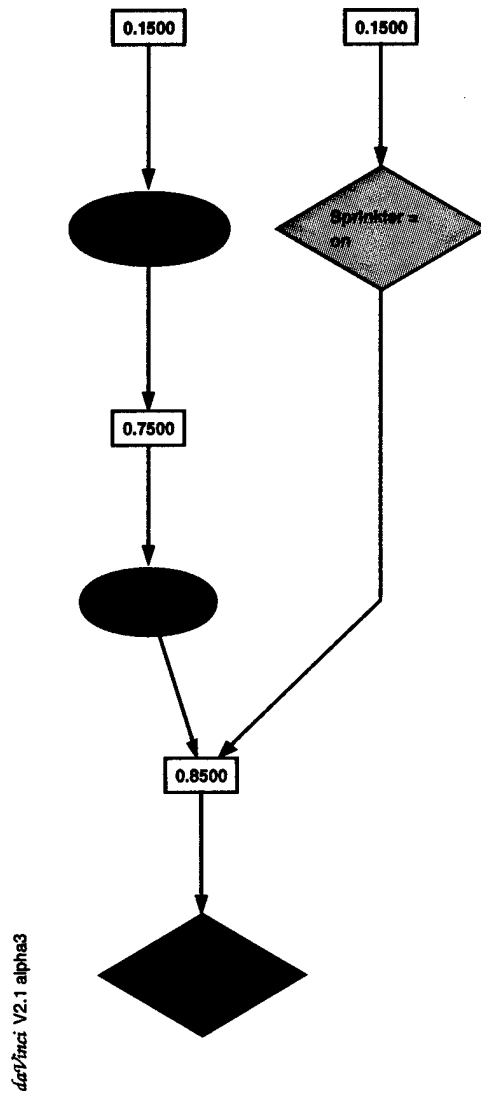
Figure 5.3   Add Mode Case 1 - Resulting BKB.

5.2 Add Mode - Case 2

The following test case is presented to PESKI:

```
Evidence is:  Sprinkler = Off
              Rain = Heavy

Answers are:  Sidewalk = Wet
```

The evidence and answer are not directly dependent upon one another due to incompleteness in the BKB as shown in figure 5.4. In this case **Sprinkler = Off** and **Rain = Heavy** are both evidence that can cause **Sidewalk = Wet**. The I-node **Sprinkler = Off** is unattached to the dependency region of **Sidewalk = Wet**. Currently, only the combination of **Sprinkler = On** and **Rain = Heavy** is currently contained in the BKB. The I-node **Sprinkler = Off** is extended and the S-node with a value of 0.75 is targeted as shown in figure 5.5. This S-node is circular in shape signifying that a new S-node will be created. The new S-node contains the desired evidence as parents. The resulting BKB is shown in figure 5.6.
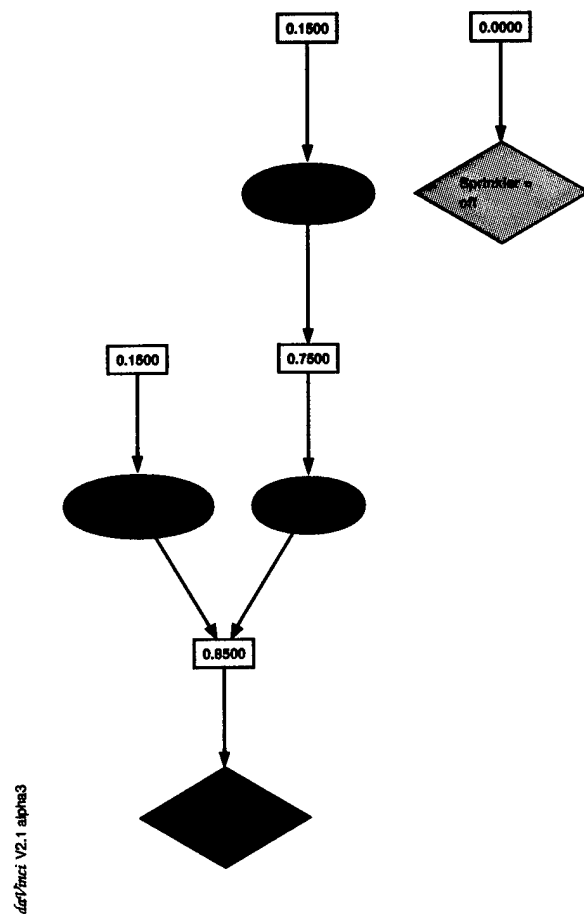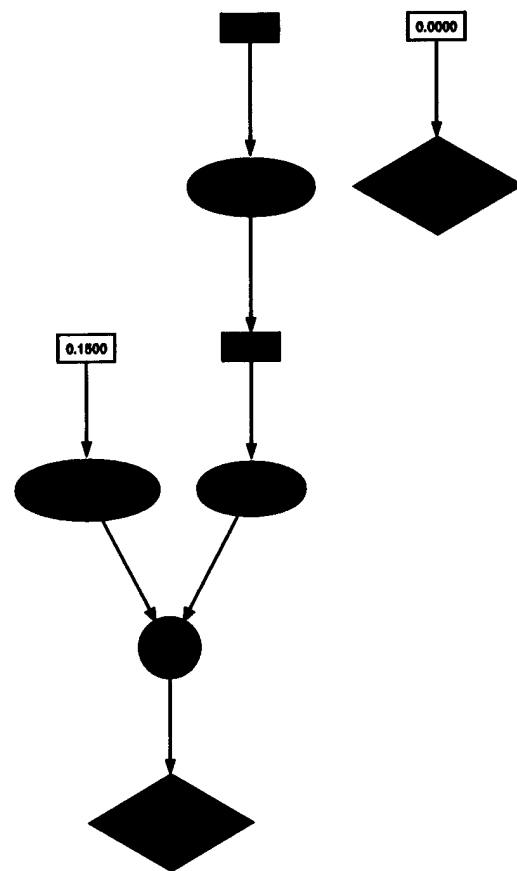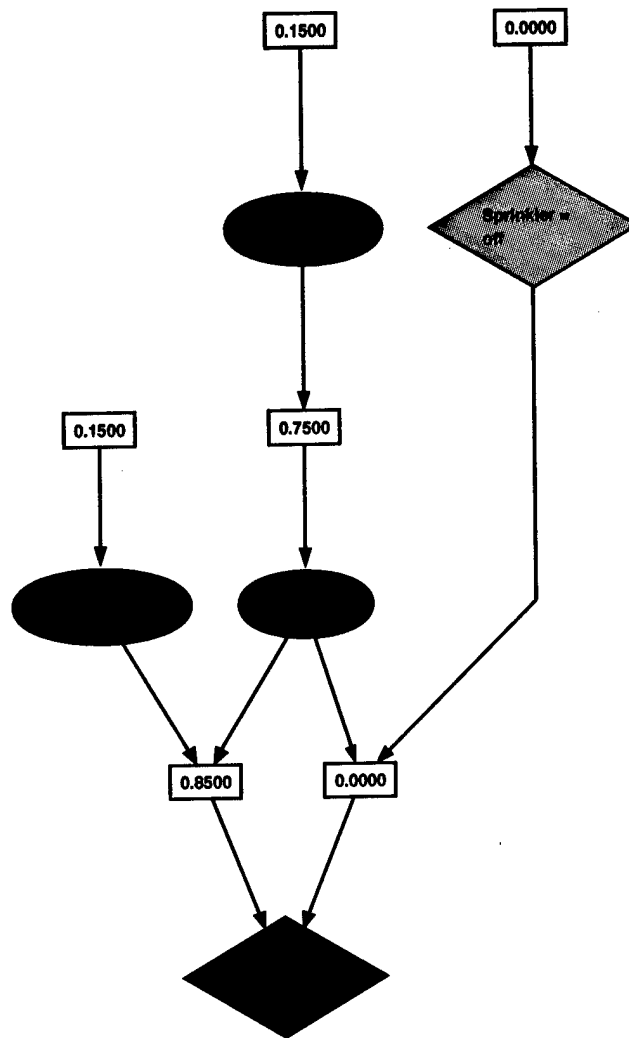
Figure 5.4   **Add Mode Case 2 .**   Evidence: Sprinkler = Off, Rain = Heavy, Answer: Sidewalk = Wet

Figure 5.5   Add Mode Case 2 - Extended I-node and Targeted S-node.

Figure 5.6   Add Mode Case 2 - Resulting BKB.

5.3 Insert Mode

The following presents an example of when the insert mode is necessary for correction of existing incompleteness. The following test case is used for the BKB shown in figure 5.7:

```
Evidence is:  Rain = Heavy

Answers are:  Sidewalk = Wet
```

When there are dark clouds in the sky, the probability of rain is often quite high. In this case the RV **Rain = Heavy** has been accidentally omitted from the dependency region of **Sidewalk = Wet**. It is intuitive that **Rain = Heavy** needs to be placed between the RVs **Clouds = Heavy** and **Sidewalk = Wet**. The user, after changing to insert mode, can then extend the I-node **Rain = Heavy** and target the S-node above **Sidewalk = Wet** as in figure 5.8. After extending, the resulting BKB is shown in figure 5.9.

5.4 Functionality of the Graphical Incompleteness Methodology

With the understanding of the functionality of the graphical incompleteness tool, the question arises whether or not this functionality allows for any addition of an incompleteness link between two nodes direct dependency regions. The functionality of the add and insert modes together have been used to recreate a highly connected BKB starting with only non-connected I-nodes. All links in this BKB were removed so that only the required single S-node existed above each of the I-nodes. The graphical incompleteness tool was then able to recreate the BKB links after receiving the necessary number of test cases. The order the test cases were given to PESKI determined the amount of insertions versus additions that were necessary. Since this BKB was able to be totally constructed using only the graphical incompleteness tool, this demonstrates that any one incompleteness link may be handled by the graphical incompleteness tool for even a highly connected BKB.
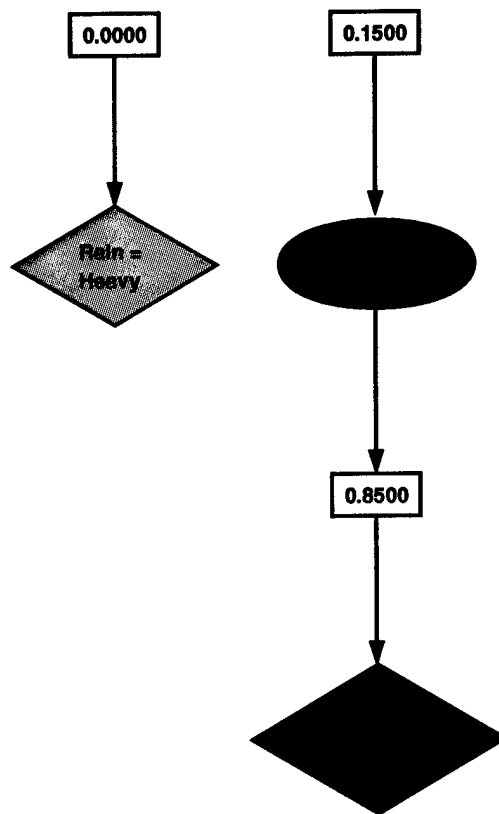
Figure 5.7 **Insert Mode .** Evidence: Rain = Heavy, Answer: Sidewalk = Wet
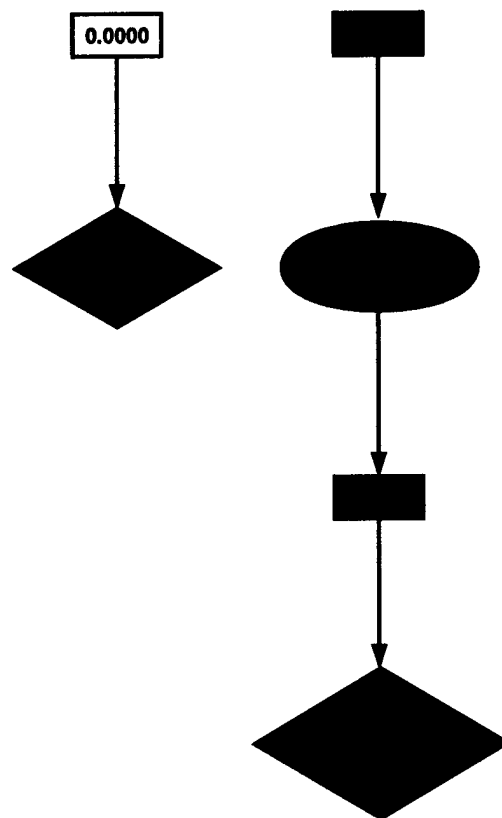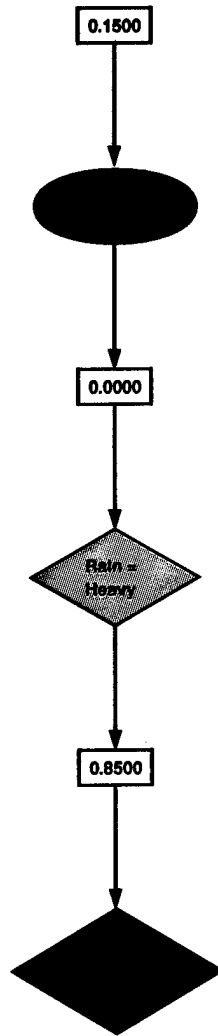
0.0000

daVinci V2.1 alpha3

Figure 5.8   **Insert Mode - Extended I-node and Targeted S-node.**

0.1500

0.0000

Rain =
Heavy

0.8500

daVinci V2.1 alpha3

Figure 5.9   Insert Mode - Resulting BKB.

VI. Conclusions

With the increasing presence of knowledge based systems, and the critical nature of many of its applications, verification and validation (V & V) of these systems is becoming increasingly important. While verification is at the forefront of currently focused research, validation methods of handling inconsistency, incompleteness, and incorrectness are scarce. PESKI is designed to be an integrated framework for the entire lifecycle of a knowledge based system using the Bayesian knowledge base (BKB) representation. This integrated framework includes V & V and the handling of these types of errors. Understanding that incompleteness in a BKB is a feature of the knowledge representation makes incompleteness handling in PESKI even more critical. The methodology developed through this research graphically presents a BKB to a knowledge engineer for the correction of incompleteness. This graphical incompleteness tool in conjunction with the data mining tool and BVAL provides a knowledge engineer the ability to efficiently validate a knowledge base. The combination of these tools provide PESKI the ability to handle the entire lifecycle of a knowledge based system; from the knowledge acquisition phase using the knowledge acquisition and/or data mining modes of PESKI, to the V & V and subsequent use of the knowledge base.

Both strengths and weaknesses of this approach were brought forth through this research. Some of the advantages noted include the following:

- The presenting of BKBs in a graphical user interface seems intuitive. The ability to quickly identify areas of direct dependency regions of both the evidence and answers from a test case can greatly aid the visualization of incompleteness. Graphical depictions of the knowledge provides explanation in the form of intermediate nodes that can be investigated for correctness and completeness. This graphical means of accessing and editing knowledge is fairly unique in the knowledge based systems field. Some commercial systems exist that allow knowledge base creation through graphical visualization, however, these systems do not extend knowledge base activity into the validation realm.

- The incremental fixing of incompleteness in test cases is achieved. As mentioned previously, the correction of one link of incompleteness can often correct other incomplete links involved in the same test case. Incremental fixing can simplify correcting test cases with multiple instances of incompleteness, rather than requiring the knowledge engineer to resolve the test case in a single step.

- The problem space is limited by displaying only direct dependency regions, which often may be small areas relative to the overall size of the BKB. The tool often made incompleteness changes fairly trivial despite a large display of nodes in some larger BKBs. Despite the large size of the BKB, after locating and extending the desired I-node, the number of possible target S-nodes were quite small and easily located and targeted. This visual assistance can really limit the amount of time that may be spent finding the incompleteness location(s).

- The manner in which the tool graphically displays the BKB and its surrounding nodes makes it possible to discover incorrectness when incompleteness may have been suspected. This detection can be very important. If automatic corrections were attempted, the system may have made non-sensical corrections and further damaged the knowledge base. When incorrectness is discovered, it can be noted and corrected in the normal knowledge acquisition mode of PESKI.

Some of the disadvantages include the following:

- Inherent with any large knowledge base is the difficulties of locating the incompleteness nodes as well as making the proper corrections for validation. The display region can, depending upon overall size as well the direct dependency region sizes of the active evidence and answer nodes, be rather large. This scalability issue will remain a problem in any type of visual format for knowledge bases. Techniques for reducing this problem would greatly assist these types of tools.

- The tool is not a "cure-all" for any incompleteness that is discovered. It may be necessary to partially correct or completely correct the incompleteness through the normal knowledge acquisition mode of PESKI. This situation occurs when the incompleteness fix demands more than the addition of a single link or the addition of

a link which ties to an S-node which currently does not exist. If the S-node does not already exist, then further incompleteness has been discovered. It can often be helpful when finding incompleteness in a test case, to further break down the test case into smaller test cases that are subsets of the previous test case. This technique can assist in both the discovering and the handling of multiple incompleteness links that may exist.

VII. Recommendations for Future Research

This chapter presents some ideas for future work related to this research area. Some enhancements to both the graphical incompleteness tool and the knowledge acquisition phases of PESKI are discussed.

7.1 Graphical Incompleteness Tool enhancement

An extension to further enhance the graphical incompleteness tool to allow multiple links of incompleteness to be added to a BKB. The tool currently allows only a connection from the direct dependency region of the evidence/answer to the direct dependency region of its corresponding evidence/answer. If the incompleteness consists of a multiple link through a node outside of either dependency region, then either the normal knowledge acquisition mode must be used to repair the incompleteness or the test case must be broken down into smaller test cases that can then be handled by the graphical incompleteness tool. An ability to allow the knowledge engineer to make these types of additions, while still ensuring that when completed that the incorrectness has been handled as well as that the BKB representation rules are still maintained, would be an added enhancement to the tool. The tool, after allowing addition of links outside of the direct dependency regions, would need to check to ensure the first and last nodes chosen were valid nodes, or would correct the incompleteness. If this were not the case, a warning message to the user would be necessary.

7.2 BKB pattern analysis

A BKB pattern analysis that can attempt to automatically correct or recommend suggestions of fixing incompleteness may be useful to a knowledge engineer, particularly for larger systems. One method of approaching this problem is to explore common instantiations of the random variables presented in the test cases. These state nodes can sometimes form similar patterns that may be detected and used to handle the incompleteness present in the current test case. These type of corrections may be automatic or textually presented to the knowledge engineer for approval before inclusion.

7.3 Tool utilization

Combining the graphical incompleteness tool with data mining could provide further assistance when attempting to fix incompleteness. When viewing the graphical incompleteness tool, it may be helpful if you could concurrently choose I-nodes and then data mine for relations between the selected I-nodes. Then by selecting the desired relationship from the data mining results, the desired incompleteness corrections would be implemented. This could make data mining more efficient by reducing the problem space.

7.4 Knowledge Acquisition Enhancements

Combining some of the ideas used in the graphical incompleteness handling methodology with the knowledge acquisition phase of PESKI could assist in easing some of the inherent difficulties of building the knowledge base. Displaying highlighted direct dependency regions as nodes are selected could assist the knowledge engineer in placing nodes and their appropriate links in their proper locations. The ability to traverse up and down the displayed nodes while not removing other nodes can be helpful as well. Currently as new nodes are requested in the graphical knowledge acquisition tool in PESKI, other nodes previously displayed are removed based upon a ranking of these nodes. Any assistance during the knowledge acquisition phase can certainly help ease the inherent difficulties of creating a knowledge base.

7.5 BKB representation enhancement

An extension to the BKB representation to allow some type of graphical or structure could assist a knowledge engineer by simplifying the graphical presentation of a BKB. The mutual exclusion constraint of the BKB representation could prove to be a limitation for knowledge acquisition. It can make knowledge base creation difficult. For example, an automobile failing to start can occur for multiple reasons like fuel system, air system, starter system, etc. These can be each independent systems, however, when placed into the BKB representation they must be tied to one another through S-nodes. This often takes the form of one system being true while all others are false. This creates difficulties for the knowledge engineer when building this type of structure in a BKB. By creating

this type of S-node, the number of links needed would be greatly reduced. This would ease visualization of the BKBs in a graphical sense as well. Certainly, all probabilities would still be necessary for input to the inference engine. The inference engine would require the ability to recognize this type of node and to inference correctly over the BKB.

Bibliography

1. Bahill, Terry A. *Verifying and Validating Personal Computer-Based Expert Systems.* Englewood Cliffs, NJ: Prentice Hall, 1991.

2. Banks, Darwyn O. *Acquiring Consistent Knowledge for Bayesian Forests.* MS thesis, AFIT/GSO/ENG/95M-01. Graduate school of engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, March 1995.

3. Charniak, Eugene. "Bayesian Networks without Tears," *AI Magazine*, 50–63 (Winter 1991).

4. Cooke, Nancy J. "Varieties of Knowledge Elicitation Techniques," *International Journal of Human-Computer Studies*, *41*(6):801–849 (1994).

5. Dempster, A. "Upper and Lower Probabilities Induced by a Multi-valued Mapping," *Annals of Mathematical Statistics*, *38*(2):325–399 (1967).

6. Duda, R., et al. "Development of the PROSPECTOR Consultation System for Mineral Exploration," *SRI Report: Stanford Research Institute* (October 1978).

7. Durkin, John. "Expert Systems: A View of the Field," *IEEE Expert*, 56–63 (April 1996).

8. Erman, D.L., et al. "The HEARSAY-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty," *Blackboard Systems*, 31–86 (1988).

9. Gleason, Howard T. *Probabilistic Knowledge Base Validation.* MS thesis, AFIT/GSO/ENG/95D-04. Graduate school of engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1995.

10. Gonzalez, Avelino J. and Douglas D. Dankel. *The Engineering of Knowledge-based Systems.* Englewood Cliffs, NJ: Prentice Hall, 1993.

11. Green, C.J.R. and M.M. Keyes. "Verification and Validation of Expert Systems," *Western Conference on Expert Systems*, 38–43 (1987).

12. Kahn, G., et al. "MORE: An Intelligent Knowledge Acquisition Tool," *Proceedings of the IJCAI-85*, 581–584 (1985).

13. Lindsay, P.H., et al. *Applications of Artificial Intelligence to Chemistry: The DENDRAL Project..* New York: McGraw-Hill, 1980.

14. Lyle, Louise J. *A Test-Case Based Approach to Bayesian Knowledge Base Incompleteness Detection and Correction.* MS thesis, AFIT/GCS/ENG/96D-17. Graduate school of engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1996.

15. Marcot, B. "Testing Your Knowledge Base," *AI Expert*, 42–47 (August 1987).

16. Marcus, S., et al. "Knowledge Acquisition for Constructive Systems," *Proceedings of the IJCAI-85*, 637–639 (1985).

17. McDermott, J. "R1: A Rule-Based Configurer of Computer Systems," *Artificial Intelligence, 19*(1):39–88 (September 1982).

18. Meseguer, P. and A. Verdaguer. "Verification of Multi-level Rule-based Expert Systems: Theory and Practice," *International Journal of Expert Systems, 6*:163–192 (1993).

19. Nazareth, Derek L. "Issues in the verification of knowledge in rule-based systems," *International Journal of Man-Machine Studies, 30*:255–271 (1989).

20. Nguyen, T.A. "Verifying Consistency of Production Systems," *Proceedings of the Third IEEE Conference on Artificial Intelligence Applications*, 4–8 (February 1987).

21. Nguyen, T.A., et al. "Checking an Expert System's Knowledge Base for Consistency and Completeness," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 375–378 (August 1985).

22. O'Keefe, Robert M., et al. "Validating Expert System Performance," *IEEE Expert*, 81–89 (Winter 1987).

23. Parsons, Simon. "Current Approaches to Handling Imperfect Information in Data and Knowledge Bases," *IEEE Transactions on Knowledge and Data Engineering, 8*:353–372 (June 1996).

24. Pearl, Judea. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Mateo, CA, 1991. (Revised Second Printing).

25. Prakash, G. Ravi and H.N. Mahabala. "SVEPOA: A Tool to Aid Verification and Validation of OPS5-based AI Applications," *International Journal of Expert Systems, 6*:193–236 (1993).

26. Preece, A.D. "Towards a Methodology for Evaluating Expert Systems," *Expert Systems, 7*:215–223 (1990).

27. Preece, Alun D. "Validation of knowledge-based systems: Current trends and issues," *The Knowledge Engineering Review, 10*(1):69–71 (1995).

28. Santos, Jr., Eugene. "A Fully Integrated Probabilistic Framework for Expert Systems Development." Research proposal from Air Force Institute of Technology to Air Force Office of Scientific Research. March 1993.

29. Santos, Jr., Eugene. "Verifying and Validating Uncertain Knowledge-Bases." Research proposal from Air Force Institute of Technology. September 1995.

30. Santos, Jr., Eugene and Darwyn O. Banks. "Acquiring Consistent Knowledge in the Face of Uncertainty," *IEEE Transactions on Knowledge and Data Engineering* (1995). (Submitted to).

31. Santos, Jr., Eugene and Sheila B. Banks. "Verifying and Validating Knowledge in the Face of Uncertainty: The PESKI System." January 1997.

32. Shachter, Ross D. "Evaluating Influence Diagrams," *Operations Research, 36*:871–872 (1986).

33. Shafer, G. *A Mathematical Theory of Evidence*. Princeton, NJ: Princeton University Press, 1976.

34. Shortliffe, E.H. *Computer-Based Medical Consultations: Mycin*. New York, NY: American Elsevier, 1976.

35. Shortliffe, E.H. and B.G. Buchanan. "A Model of Inexact Reasoning in Medicine," *Mathematical Biosciences*, *23*:351–379 (1975).

36. Stachowitz, R., et al. "Building Validation Tools for Knowledge-Based Systems," *First Annual Workshop on Space Operations Automation and Robotics (SOAR 87)*, 7:209–216 (1987).

37. Stein III, Daniel J. *Utilizing Data and Knowledge Mining for Probabilistic Knowledge Bases*. MS thesis, AFIT/GCS/ENG/96D-25. Graduate school of engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1996.

38. Zadeh, L.A. "Fuzzy Sets," *Information and Control*, *8*(3):338–353 (1965).

Appendix A. PESKI

PESKI is an integrated framework for the development of knowledge based systems. The Bayesian Knowledge Base is a key element of the expert system architecture called PESKI (Probabilities, Expert System, Knowledge and Inference)[28].
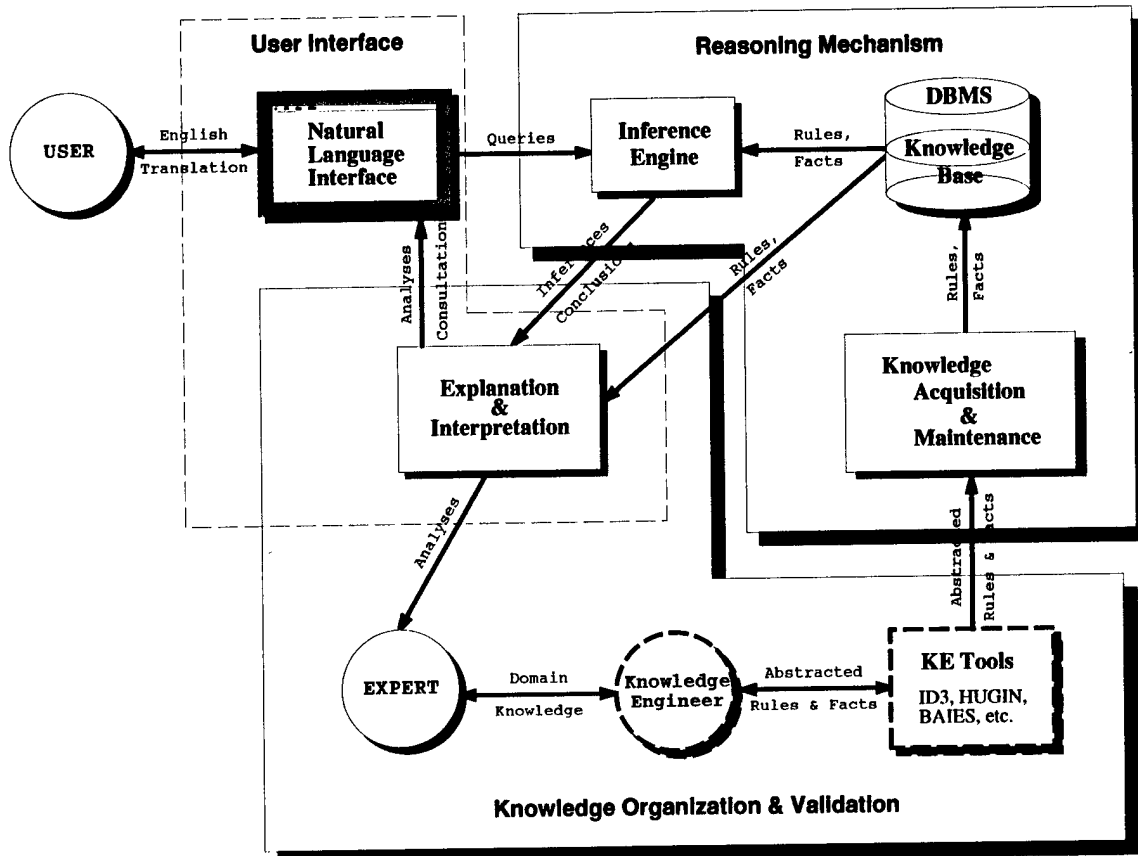

Figure A.1   **The PESKI architecture.**   The broken boarder components **Knowledge Engineer** and **KE Tools** are considered optional.

As shown in Figure A.1, the PESKI architecture is composed of four major components:

- **Natural Language Interface**—provides for user–system communication by translating user queries, and system responses, into English.

- **Inference Engine**—the mechanism responsible for the reasoning actions of the system, controlling choice and application of information contained in the knowledge base.

- **Explanation & Interpretation**—tracks paths used by the inference engine in reaching its conclusion.

- **Knowledge Acquisition & Maintenance**—provides the tools for incorporating new or updated expert knowledge into the knowledge base.

There is some measure of overlap between these four components, hence PESKI facilitates the combination of these components into three overlapping subsystems:

- *User Interface*—composed of the Natural Language Interface and the Explanation & Interpretation components.

- *Knowledge Organization & Validation*—consists of the Explanation & Interpretation component along with the human expert, the optional knowledge engineer, and the knowledge engineering tools. Communication with the the Knowledge Acquisition & Maintenance component facilitates organization, and then assists in Validation when coupled with feedback from the Reasoning Mechanism through Explanation &Interpretation.

- *Reasoning Mechanism*—combines the Inference Engine and the Knowledge Acquisition & Maintenance components. The merging of these two components stems from the belief that in order for new knowledge to be placed in the knowledge base, some form of reasoning (and possibly learning) must be involved. Additionally, the inclusion of the Knowledge Acquisition & Maintenance component provides some degree of information hiding with respect to the knowledge representation used with the knowledge base.

The PESKI architecture is sufficient for the construction of knowledge-based systems in nearly any domain.

Appendix B. BKB Incompleteness Relative to Bayesian Networks

The total amount of global incompleteness in a BKB, whether desired or in error, can be related to the amount of necessary information in a Bayesian network. Again, a Bayesian network requires complete conditional probability tables (CPTs) between linked nodes to inference over the network. In a BKB, an S-node represents one CPT entry in a Bayesian network. The amount of incompleteness in a BKB relative to a complete Bayesian network can therefore be determined with the following formula:

$$\text{Amount of Incompleteness} = 1 - \frac{\text{Number of S-nodes}}{\sum_{i=1}^{N} |CPT_i|}$$

where N = Number of RVs

and $|CPT_i| = (\prod_{j=1}^{M} \text{Number of states of Parent } RV_j) * \text{Number states of } RV_i$

where M = Number of parents of RV_i

This formula can be used to consider the results obtained through the different approaches used in the graphical incompleteness handling methodology. In add mode Case 2, the addition of a new S-node and its appropriate links is similar to filling in a missing item in the CPT of a Bayesian network. Therefore, the total amount of incompleteness in the BKB relative to a complete Bayesian network is always decreased with this type of correction. However, in add mode Case 1, conditional probability tables are increased in size in addition to filling in a missing probability. This correction will result in an increase in overall incompleteness; however, validation of BKBs is more concerned with relevance and correctness of the fix more so than the amount of incompleteness relative to a Bayesian network. The increased amount of incompleteness may not be relevant to the inferencing necessary for the system, therefore the amount of incompleteness that can be considered mandatory for the system very well may have decreased. Insert mode is concerned with the restructuring of the BKB rather than a decrease in overall incompleteness. Again, correctness of the BKB is of more concern than the amount of incompleteness relative to a Bayesian network.

Vita

David Bawcom was born in New Orleans, LA in August, 1968. He was awarded a Bachelor of Science in Electrical Engineering from McNeese State University in December, 1990. While pursuing his degree and until he joined the U.S. Air Force through Officer Training School in January, 1995, he was a member of the U.S. Navy Reserves. His first assignment with the Air Force was to the Air Force Operational Test and Evaluatation Center where he did modeling and analysis for a number of different test programs. He arrived at the Air Force Institute of Technology in May 1996.

Permanent address: 4211 Phillips St., Enon, OH 45323

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1997	Master's Thesis

4. TITLE AND SUBTITLE
An Incompleteness Handling Methodology for Bayesian Knowledge Bases

5. FUNDING NUMBERS

6. AUTHOR(S)
David J. Bawcom, First Lieutenant USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GCS/ENG/97D-02

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
HQ AFMC/CI
WPAFB OH 45433

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
The PESKI (Probabilities, Expert Systems, Knowledge, and Inference) system attempts to address some of the problems in expert system design through the use of the Bayesian Knowledge Base (BKB) representation. Knowledge gathered from a domain expert is placed into this framework and inferencing is performed over it. However, by the nature of BKBs, not all knowledge is incorporated, i.e. the representation need not be a complete representation of all combinations and possibilities of the knowledge, as this would be impractical in many real-world systems. Therefore, inherent in such a system is the problem of incomplete knowledge, or spaces within the knowledge base where areas of lacking knowledge hinder arrival at a solution. Some of this knowledge is intentionally omitted but necessary for valid results. Intentional omission, a strength of the BKB representation, allows for capturing only the relevant portions of knowledge critical to modeling an expert's knowledge within a domain. This research proposes a method for handling the latter form of incompleteness administered through a graphical interface. The incompleteness is then able to be detected and corrected by the knowledge engineer in an intuitive fashion.

14. SUBJECT TERMS
Knowledge engineering, Expert systems, Incompleteness handling, Bayesian Knowledge Bases

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL